2015-09

GAPR2: a DTN routing protocol for communications in challenged, degraded, and denied environments

Killeen, Kevin M., Jr.

Monterey, California: Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

GAPR2: A DTN ROUTING PROTOCOL FOR COMMUNICATIONS IN CHALLENGED, DEGRADED, AND DENIED ENVIRONMENTS

by

Kevin M. Killeen Jr

September 2015

Thesis Advisor: Justin P. Rohrer
Second Reader: Geoffrey G. Xie

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2015	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE GAPR2: A DTN ROUTING PROTOCOL FOR COMMUNICATIONS IN CHALLENGED, DEGRADED, AND DENIED ENVIRONMENTS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Killeen, Kevin M. Jr	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

This thesis explores the foundation of modern Disruption Tolerant Protocols. It introduces a previously unpublished protocol with high probability of delivery called the Geolocation Assisted Predictive Routing (GAPR) Protocol and implements Vector Routing for The ONE Simulator. GAPR and Vector are then combined and implemented as GAPR2, a new protocol that provides delivery ratio near the best in the field while incurring a quarter of the overhead. GAPR2, GAPR, and Vector, along with the most widely referenced DTN routing protocols (Epidemic, MaxProp, and PRoPHETv2) are extensively simulated and their performance evaluated using three mobility models: the Helsinki scenario, a random mobility model, and a military scenario based on a real-world annual exercise. The custom-built military mobility model uses GIS topographical data and custom GIS overlays to implement a realistic scenario terrain. The performance of each protocol is evaluated.

This thesis shows through simulation that DTN networks can be employed to enhance communications capabilities without expensive infrastructure or significant platform upgrades. Further, this thesis shows through large-scale simulations that such a network is capable of operating over hundreds of square kilometers and provides the simulation framework to test future routing protocols or equipment configurations.

14. SUBJECT TERMS networking, DTN, DTN routing, GAPR, GAPR2, MaxProp, PRoPHET, Vector, DTN simulation, ONE, ONE Simulator, DTN mobility, DTN simulation using GIS	**15. NUMBER OF PAGES** 139
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

GAPR2: A DTN ROUTING PROTOCOL FOR COMMUNICATIONS IN CHALLENGED, DEGRADED, AND DENIED ENVIRONMENTS

Kevin M. Killeen Jr
Lieutenant, United Stated Navy
B.S., United States Naval Academy, 2009

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2015

Author: Kevin M. Killeen Jr

Approved by: Justin P. Rohrer
Thesis Advisor

Geoffrey G. Xie
Second Reader

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis explores the foundation of modern Disruption Tolerant Protocols. It introduces a previously unpublished protocol with high probability of delivery called the Geolocation Assisted Predictive Routing (GAPR) Protocol and implements Vector Routing for The ONE Simulator. GAPR and Vector are then combined and implemented as GAPR2, a new protocol that provides delivery ratio near the best in the field while incurring a quarter of the overhead. GAPR2, GAPR, and Vector, along with the most widely referenced DTN routing protocols (Epidemic, MaxProp, and PRoPHETv2) are extensively simulated and their performance evaluated using three mobility models: the Helsinki scenario, a random mobility model, and a military scenario based on a real-world annual exercise. The custom-built military mobility model uses GIS topographical data and custom GIS overlays to implement a realistic scenario terrain. The performance of each protocol is evaluated.

This thesis shows through simulation that DTN networks can be employed to enhance communications capabilities without expensive infrastructure or significant platform upgrades. Further, this thesis shows through large-scale simulations that such a network is capable of operating over hundreds of square kilometers and provides the simulation framework to test future routing protocols or equipment configurations.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgements

My thanks go out to my thesis advisor, Professor Justin Rohrer, whose class on Network Modeling first piqued my interest in DTN research, and who agreed to sign on as my advisor even though I had a very late start on this thesis. Our weekly meetings are the only reason I managed to finish this thesis before graduating. My thanks also go out to professor Geoffrey Xie, my second reader and a member of the group that first developed the GAPR protocol, along with Professor Rohrer and Professor Robert Beverly.

I am grateful for the opportunity provided by Naval Postgraduate School to tackle a technical masters program after a undergraduate BS in Political Science and to all the professors who helped me succeed in a new curriculum. Finally, I need to thank the other students in my cohort, the friends I made in Monterey, and my family, all of whom provided the motivation to complete this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: INTRODUCTION

Disruption Tolerant Networks (DTNs) are an exciting new class of networks that support infrastructureless mobile ad-hoc networksand do not require an end-to-end path to begin forwarding messages towards a destination. This is a radical departure from traditional network architecture, and DTN researchers are actively developing and implementing protocols that can support message routing over networks without end-to-end paths.

The field of research is just over a decade old, and a quick scan of the published DTN literature shows that the field is growing rapidly. Indeed, the field has expanded so fast that research into many applications for DTN technology is ongoing, and no complete survey of the field or unified taxonomy has been universally accepted. The focus of this chapter is introducing our research project. We first provide a more complete definition of the term DTN, then discuss our objectives, define the scope and limitations of this project, and finally summarize the structure of this thesis. Chapter two presents a more cohesive background and a more detailed description of how DTN routing works; it is recommended that readers conducting background research on DTNs but not interested in this particular project begin by reading Chapter 2.

1.1 Definition of a DTN

A graph, or network, is defined by a set of vertices (nodes), and a set of edges (links), where the set of edges form some subset of the set of all possible pairs of nodes. Thus a network can be thought of as a group of nodes where some nodes are connected to other nodes by edges. The number of edges in this graph can thus vary from zero to one less than the square of the number of nodes; the greater the number of edges, the more connected the network.

A classical, hierarchical communications network is often defined in similar terms. In such networks the nodes on the exterior are typically hosts and nodes at the interior of the network are routers. The hosts generate and receive messages, and the routers forward the messages, or traffic, between the hosts. The edges that connect the hosts and routers represent the communications links between the nodes.

The entire reason these networks exist, at the most basic level, is to provide a means by which a source node can send messages to a destination node. Networks accomplish this by advertising

the paths between nodes, so that hosts are able to identify destinations on the network that are reachable. When the source is ready to transmit a message, a subset of the nodes and edges that connect the source to the destination are selected to form a path between the two nodes. Messages are sent along this path, with every subsequent node forwarding the message to the next link in the path until the message arrives at the destination. The route from the source to the destination is called an end-to-end path. All widely-deployed communications networks including the Internet, cellphone system, and even satellite networks require an end-to-end path before messages can be forwarded. This means that intermittently-connected nodes or hosts without access to the infrastructure used to establish consistent end to end connections are unable to communicate.

These networks rely on a consistent topology with a large number of permanant links to create stable paths. If the network topology changes frequently, then paths that exist when a message leaves a source can change or break entirely while the message is still en-route. If the routers along that path can no longer connect the source to destination because of the changes, the message can not be delivered. When this happens on the Internet, the message is dropped and the user receives an error: destination unreachable message (if the source can still be reached). Likewise, too few links can cause a partitioned network where only small groups of nodes can communicate, but nodes in one group cannot reach nodes in the other group.

Delay Tolerant Networks are a revolutionary new class of networks that approach routing in a fundamentally differently way then the traditional network protocols described above. Instead of reliance on coherent end-to-end paths from the source to the destination before message routing can begin, DTN protocols leverage opportunistic routing mechanisms to take advantage of connections when they are available. In mobile DTNs, the focus of this work, opportunistic routing also naturally leverages the Store-Carry-Forward paradigm by using the mobility of intermediary 'carrier' nodes to facilitate routing across networks where end-to-end paths are never formed.

Under the Store-Carry-Forward paradigm, the source node forwards messages to other nodes, which then store the messages in a buffer. The node that received and stored the message then carries that message as it moves, forwarding it to other nodes along its path. Eventually, the message will propagate through the network and reach the destination. DTNs are also typically peer-to-peer networks; participating nodes work together to forward each other's traffic. In this way, the distinction between hosts and routers evaporates, with each participating node fulfilling

both roles. Finally, DTN nodes join, organize, and connect to other nodes autonomously on an ad-hoc basis. The Store-Carry-Forward mechanism and other characteristics of DTNs are discussed in more detail at the beginning of chapter two.

1.2 Objectives

DTN routing is a very active area of research, with dozens of protocols proposed, and even more modifications suggested to improve the performance of top-performing protocols. There is no consensus on the best routing protocol, because the performance of a given protocol is highly dependent on the characteristics of the network in which it is implemented. In recent years a few protocols have come to dominate mainstream DTN research efforts, yet some promising earlier research pointed to inventive ideas for making routing decisions has been largely overlooked. In this project we revisit two routing protocols designed to address specific situations that tend to degrade DTN performance. We first attempt to recreate the protocols and validate the original developer's performance metrics through simulation. We then combine the two protocols, forming a new blended protocol we call GAPR2, which is a compromise between the high delivery ratio of one and the low overhead of the other. We determine through simulation how GAPR2 performs as compared to the original protocols and the leading protocols in the field.

In 2008, Vector Routing for Delay Tolerant Networks [1] was published. This protocol attempts to limit the number of replicated messages in flooding-based protocols by controlling the maximum amount of messages that can be exchanged during an encounter. The degree of replication is made proportional to how orthogonal the angle of incidence between the two node's momentum vectors is at the start of an encounter. In the simulation conducted by the authors, Vector routing outperformed epidemic routing and significant decreases in overhead and buffer growth were noted. However, the simulation work was limited, and the Vector logic has not been widely researched.

Likewise, in 2012 a draft DTN protocol was completed called the Geolocation Assisted Predictive Routing protocol (GAPR) [2]. GAPR was an attempt to leverage the best aspects of the state of the art protocols with a positional awareness mechanism to prevent poor routing decisions caused by outdated information. In initial simulations during the original research GAPR performed as well as or better than competing DTN routing protocols including MaxProp and PRoPHETv2, the two most widely-researched DTN protocols. However, extensive simulation was never completed and the project was halted.

1.2.1 Simulate Original Protocols and Validate Past Work

The first objective of this thesis is to validate the results from the original work introducing both the Vector and GAPR protocols. The designers of both protocols used simulation to benchmark these protocols against other popular DTN protocols. Therefore, the first step in an effort to reintroduce these protocols and to build upon them is too implement both protocols in a single simulator and verify that each protocol is performing as described in the original work. This will allow side-by-side evaluation against well-known protocols similar to the original trials, and will also allow Vector and GAPR to be compared directly against one another.

GAPR was originally simulated using the opportunistic network environment simulator (The ONE Simulator, or ONE) [3], whereas Vector routing was originally simulated using ns-2 [4]. ONE is used with the original source code from the GAPR project, and a new simulation file is built to model Vector routing in ONE. It is essential to note that [1] does not include source code, and does not discuss aspects of the protocol implementation beyond the vector building and weighting algorithm. Further discussion of choices made when implementing vector routing in ONE can be found in the methodology chapter (Chapter 3.2).

The Helsinki Simulation is used for the validation trials. The GAPR validation simulations are identical to those in the original publication, and confirm the findings of the authors. Vector routing was more difficult to validate because the exact implementation of the original protocol is unknown and this work uses a different simulator. Instead, a combination of code logs and analysis was used to validate operation of Vector. Validating past work also ensures that updates to the original GAPR code for compatibility with ONEv1.5.1 did not effect the performance of the protocol.

1.2.2 Combine Vector and GAPR

GAPR and Vector both rely on positional awareness to improve message flooding choices during node encounters. However, GAPR uses positional awareness to reduce the likelihood of miscalculating delivery probabilities, whereas Vector uses each nodes own positional history to limit the number of exchanged messages during an encounter based on the difference of the nodes movement. This thesis combines the limited flooding mechanism of Vector routing with the delivery probability and positional awareness mechanisms in GAPR to produce a new protocol, GAPR2. This protocol is also a simulated alongside and analyzed against the field of DTN protocols.

1.2.3 Extensive Simulation of Vector, GAPR, and GAPR2

The simulation work included in the original publications of Vector [1] and GAPR [2] was limited. This thesis expands the original work by conducting extensive simulation and comparison against the most prolific DTN routing protocols. This work includes simulation across a wide range of variables and mobility models, and provides comprehensive performance data on each protocol. It includes more analysis and figures generated from a larger range of simulations then was possible in the original publications, across simulations of the Helsinki Scenario, a random mobility scenario, and a custom scenario with a military-oriented mobiltiy model.

1.2.4 Simulate Performance during a Military Operation

DTN technology is uniquely well-suited to provide enhanced communication capabilities during deployed military operations. However, no open-source simulator or public research literature includes discussion of a mobility model that is developed based on a military operation. Therefore, a simulation based on the Bold Alligator exercises [5] is developed. Vector, GAPR, and other DTN protocols are then simulated using this mobility model to gain insights into the performance of these protocols in a realistic military-oriented operation.

1.3 Scope

The scope of this thesis is limited to a proper subset of the field of DTN research dealing with intermittently connected mobile disruption tolerant networks. Only six routing protocols are simulated in this work, but those six include the three most popular DTN routing protocols along with the two protocols this work seeks to more exhaustively simulate, GAPR and Vector, and a new protocol that combines the logic of both, GAPR2. Chapter two includes background information necessary for the average reader to understand the methodology, simulations, and conclusions reached in this paper. However, since the focus of this work is limited to routing, other aspects of the DTN archetecture including bundling protocols, gateways, and integration with the Internet are not discussed in depth. Other publications on these topics are widespread; for those interested, two papers written by one of the original designers of DTN architecture, Kevin Fall, provides an excellent introduction: [6], [7].

The publication used to introduce the Vector routing protocol only includes a description of the algorithm and pseudo code. Further, the initial simulation for vector routing was done in ns-2, and the original source code was not obtainable, nor would it compile in ONE. Therefore, the vector routing implementation used in this thesis is guided by [1], but the Java-based

implementation in ONE is original.

DTN routing has also garnered the interest of many researchers working on autonomous sensor nets. Sensor net DTNs are not considered specifically, as this work considers routing approaches and is agnostic to the application layer. When the term DTN is used in this paper it is implied to specifically refer to delay and disruption tolerant intermittently connected mobile ad hoc networks. A brief discussion of taxonomy is included in Chapter 2.1.4, where terminology is further clarified.

1.4 Limitations

The first step of this experiment is to validate the results obtained during the original work on GAPR and Vector. However, between the time that GAPR was developed and these experiments were conducted both The ONE Simulator and the Java environment in which ONE runs were upgraded to new versions. The original work done with GAPR and Vector is validated at the start of chapter four, but it is possible that there may be minor deviations in long simulation runs due to these updates. We believe any effect caused by these changes is negligible, and no significant impact was noted during the analysis of simulation data. MaxProp and PRoPHETv2 are included in the source package with ONE version 1.5.1; the performance of these protocols appears to be in line with other published work, but the implementations of these protocols in ONE are not independently verified for this thesis.

The custom military-oriented mobility model used for this experiment is based on an annual major joint exercise called Bold Alligator [5]. Only publicly disseminated data such as the units participating, the objectives, and the general plans for the exercise are used to inform the scenario, and there is not sufficient data to implement the exercise exactly as it occurred. Further, the level of realism and data required to implement such a realistic simulation is beyond the scope of this thesis. Instead, the military mobility model is developed using the plans and press releases for Bold alligator 2012 as guidelines. While it would be preferred to have used actual track data taken from participating units to build this mobility model, given the time frame for this thesis such an effort is not a realistic option. Regardless, we believe that the mobility model used represents a realistic, high-fidelity military operation and conforms to the real-world planning guidance of the Bold Alligator exercises.

Finally, thousands simulation runs were conducted over the course of this work, with each run capable of producing dozens of reports, and each report representing dozens of data points.

Such an overflow of data would be unworkable, therefore this work looks only at the summary statistics reports for each simulation run. Even the raw data from the summary reports represents tens of thousands of individual measurements. Thus, the raw data is not included, but parsed and aggregated data is included as appendicies A-E.

1.5 Structure of the Thesis

Chapter 2 examines the precursors to modern DTN's, including the Interplanetary Internet and mobile ad hoc networks, since it is within these precursor technologies that the foundations of modern DTNs were laid. This is followed by a characterization of typical DTN behavior, and several examples of real-world projects that leveraged this technology. Current interest in DTN technology across the public, private, and military spheres is discussed as well, along with how DTNs will continue to influence technology into the future, all justifies the need for continued DTN research and this thesis. Additional background information regarding DTN routing, including the underpinnings of various routing strategies and behaviors, and the metrics by which protocols are compared, is also summarized. Finally, each protocol that is simulated is described, and a review of relevant literature is conducted.

Chapter 3 describes the methodology this thesis follows, including how the simulator operates and how GAPR2 and Vector are implemented in the simulator. A description of each scenario and the constraints that the scenario places on simulation variables is then presented. Finally, each set of simulations is outlined, including tables that show the settings for each set of simulations.

Chapter 4 presents the data derived from the simulations that are described in chapter three, and provide analysis of the simulations. This data and analysis informs several conclusions presented in chapter 5. Chapter 5 also includes recommendations for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background

DTN research is a relatively young field which already offers unprecedented networking capabilities that are currently being leveraged in revolutionary ways to improve the capabilities of researchers and the lives of people in undeveloped areas. In the future flexible and adaptive DTN protocols will be an enabler of greater connectivity, paving the way for unprecedented expansions of the Internet of Things. This chapter offers background on DTN development, including major precursor technologies that paved the way for modern DTN research, a definition of what is meant by "modern DTN," and the characteristics of such a network. Current applications of these networks and interest in future applications of the technology, particularly for military uses, are then presented both to justify this work and to show the need for additional research into enhancing DTN capabilities.

The focus of this work is on DTN routing protocols, thus the second half of this chapter focuses on providing additional background information specifically on routing protocols. First is a discussion of the primary attributes that determine the most appropriate routing strategy and a presentation of those strategies. The measures of performance by which protocols are evaluated and compared are then explained, followed by a description of the specific protocols in this work. Finally, a review of similar literature to this work is presented.

2.1 History of Development

The first modern DTN protocol was published in 2000 by Vahdat and Becker in Epidemic Routing for Partially Connected Ad-hoc Networks [8], [9]. At the same time, the first widely referenced DTN implementation in a real-world network was underway in Africa, where researchers used a direct contact DTN to track wildlife across the African wilderness in the ZebraNet project [10]. However, a 2003 paper presented at the ACM SIGCOMM conference by Kevin Fall titled A Delay-tolerant Network Architecture for Challenged Internets coined the term DTN, and is generally credited as the first cohesive presentation of a modern DTN architecture [6].

Fall suggested that the autonomous organization and routing capabilities of mobile ad-hoc networks and the opportunistic store-carry-forward mechanism developed by the Interplanetary Internet Project, now the Interplanetary Networking Special Interest Group (IPNSIG), could be

leveraged to build a general-purpose terrestrial delay tolerant network architecture. In a later journal publication recounting the evolution of DTN technology, Fall writes, “at its inception, the concepts behind the DTN architecture were primarily targeted at tolerating long delays and predictably-interrupted communications over long distances... The DTN architectural emphasis [has grown] from scheduled connectivity in the IPN case to include other types of networks” [7].

This foundational paper referenced three precursor technologies: the delay tolerance developed to support satellite communications and the deep space network, the opportunistic store-carry-forward paradigm used in the Interplanetary Internet, at the mobility tolerance and ad-hoc organization of mobile ad hoc networks. These technologies are the building blocks of all DTN routing protocols discussed in this thesis.

2.1.1 Space Networks and the Interplanetary Internet

Most Network protocols and applications have developed without a high tolerance for delay or disruption. The most recognizable set of protocols, the TCP/IP stack that the modern Internet is built on, relies on a fairly static highly-connected infrastructure. The primary routing protocol in the TCP/IP stack, the Transmission Control Protocol (TCP), is one example of a transport protocol whose performance quickly degrades in the presence of significant delay.

To provide dependable transport, TCP creates a virtual connection between two hosts that is synchronized and accepted by both ends of the connection. Data sent over this virtual connection is acknowledged by the destination before the source sends additional data. This results in a back-and-forth exchange of data and acknowledgements. Those exchanges, along with TCP congestion control mechanisms, make the protocol untenable over delay-laden links. However, with the advent of satellite communications and networking, a reliable transport protocol was required that could function over connections with orders of magnitude more propagation delay then wired networks on earth.

Recognizing the need to address the shortcomings of TCP when used over networks with large propagation delays, the Internet Engineering Task Force (IETF) stood up a working group to recommend configurations and modifications that enhance TCP’s performance over satellite links; the adaptation of TCP for satellite is detailed in RFC 2488, and known colloquially as TCPsat [11]. Most of the alterations aim to reduce the number of round trips required to transmit data, such as TCP extensions for transactions (T/TCP) wherein the syn and the first TCP segment are sent together. Alterations to congestion control mechanisms, particularly

reducing the effects of slow start by increasing the initial value of cwind and using partial ACKs, also aim to reduce the number of round trip times and unnecessary retransmissions. Explicit Congestion Notification (ECN) is an alternative TCP mechanism for congestion control where the routers notifies the sender of congestion, and the sender slows transmission to avoid overflowing router buffers. This also results in fewer lost packets, thus fewer retransmissions and fewer round trips. Delay tolerant network routing protocols do not typically provide reliable transport. However, use of acknowledgements and efficient use of buffer space is also essential to DTN performance [11], [12].

Where propagation delay can be addressed through modifications to traditional protocol logic, another source of delay in space-based communications is caused by the physical sparsity of nodes. Satellite networks and deep space probes cannot count on always having intermediary nodes (other satellites) available to form an end-to-end path from a satellite to the ground-station destination. Other satellites in the network can be out of range or obstructed by planetary bodies, or the ground station may not be in a position that allows it to connect to the satellites. This often results in the inability for a source to form a path to the destination. Traditional Networks require that an end-to-end path from the source to destination is established before message routing can commence.

Satellite networks have traditionally handled this problem at the application layer. Engineers calculate when the satellite will be in a position to contact the ground center directly, or to forward messages through other satellites that can forward those messages to the ground center. This concept of programming nodes to transfer to other intermediaries which then store the messages and carry them to another pre-calculated connection opportunity, eventually relaying the messages to the destination, is called store-carry-forward. In one sense, these networks do not require contiguous end-to-end path, marking a vast departure from traditional routing. However, these networks actually do have a end to end path pre-calculated when the message leaves the source node. Furthermore, this behavior is transparent to the routing protocol, as it's handled at the application layer; as far as the routing protocol is concerned, each hop between satellites is a single-hop route. This is a concession to allow easy compatibility with traditional routing protocols and the TCP/IP stack. However, in a revolutionary step forward, a group of engineers at NASA working on a new space communications project, the Interplanetary Internet, took the concept of store-carry-forward and applied it to a opportunistic encounter scheme.

The Interplanetary Internet project began in 1998 with the ultimate goal of developing specifications for a network of nodes located on different planets of the solar system. In the near-term, the team aimed to provide richer, Internet-like connectivity to space probes, rovers, and exploration vehicles, focusing first on Mars. However, the TCP/IP stack is poorly suited to support communication between ground stations on earth and ground rovers on Mars. The propagation delay between earth and Mars is between 7 and 40 minutes depending on the position of the two planets in their orbits. Additional variable delays caused by the rotation of the planetary bodies and orbit of intermediary satellites are also present.

The first Mars Rover, Sojourner, was designed to transmit directly back to the NASA control center. However, the alignment of the planets severely limited the duration of Sojourner's communications window with Earth. To overcome this problem, NASA developed a plan by which Sojourner and future nodes on the surface of Mars transmit their data to satellites orbiting Mars. The satellites store that information until a connection can be made to the deep space communication network (DSN). They then forward that information to the DSN, which operates as a standard satellite communications network. Importantly, Mars Rover's don't wait until a scheduled contact period to forward messages to the Mars Satellites; the connect opportunistically whenever a connection can be made and there is data to forward. The Mars Satellites work the same way in connecting to the DSN.

NASA is now working to standardize these protocols across all exploration craft in order to create a true network of surface nodes and orbiting satellite nodes that operate on a store and forward basis [13]. The routing protocol that runs the store and forward mechanism used by the Interplanetary Internet is called the Licklider transmission protocol, specified in IETF RFC 5326 [14]. From an IEEE spectrum article written by Joab Jackson on how the Interplanetary Internet works [15]:

> Suppose a robotic surveyor on Mars has to navigate harsh terrain, looking for rocks that might contain fossils, and then send new photos of them back to Earth–a 10- to 12-minute trip at best. If it were a node on a TCP/IP network, the robot would have to keep a copy of that data in its limited memory banks until it got a confirmation that the data had been received on Earth. Such a notice would take at least 20 minutes to arrive–more if a direct connection weren't available. DTN, on the other hand, would require the surveyor to keep the data only until they were received by the first node–probably a nearby relay satellite. The surveyor could empty its

memory banks and go back to snapping more photos within seconds.

For additional reading: [16].

2.1.2 Mobile Ad-Hoc Networks

DTNs also leverage research into Mobile Ad-Hoc Networks (MANETs), where nodes are self-organizing and mobile. MANET nodes can join and leave a network at-will, and are autonomously configured when entering. Nodes generally exhibit some degree of random mobility, breaking and establishing new links autonomously as they move. A key feature of a MANET is to degrade gracefully [17]; when nodes are highly connected performance should not suffer from burdensome overhead and control messages, and when connections are scarce the network continues to support nodes that remain connected and allow disconnected nodes to rejoin. "In ad hoc networks, the devices themselves are the network, and this allows seamless communication, at low cost, in a self-organizing fashion and with easy deployment" [18].

Routing in MANET is complicated by the temporary ad-hoc nature of the links. Some MANET protocols deal with rapidly changing network topologies by attempting to build end-to-end routes reactively when a packet needs to be forwarded, called source-initiated, demand-driven, or on-demand routing. Others work pro-actively to build routes ahead of time and forward status messaged throughout the network; this approach is known as table-driven routing [19]. Regardless of approach, the objective is to form a end-to-end route so messages can be forwarded, and message forwarding in a MANET requires end-to-end paths.

2.1.3 Disruption Tolerant routing

DTN routing protocols draw from the autonomous organizational behavior of MANET protocols, implement the disruption and delay tolerance of space-based networks, and make use of the opportunistic store-carry-forward paradigm developed for the Interplanetary Internet. However, these routing protocols are forced to deal with a strikingly hard challenge: how to route a message when there's scant information to inform nodes of the best intermediaries through which to route a message. In the case of the IPI, there are limited options at each stage in the path–from a Mars rover to a orbiting satellite to The DSN to a Earth satellite to a base station. And in other classes of networks, end-to-end path requirements negate this issue entirely.

The strategy underlying how intermediaries are chosen reside in the routing protocols; they are responsible for determining the best available method of reaching the destination with whatever

limited information is available to the node in question. DTN performance is most directly tied to routing protocol performance because of this.

2.1.4 Characteristics of the Modern DTN

The taxonomy of disruption tolerant networking research is still evolving. What are colloquially referred to as disruption tolerant networks, both in this thesis and in the literature of this field at large, are more specifically termed delay and disruption tolerant intermittently connected networks. Such networks are a combination of the characteristics of intermittently connected networks and delay tolerant networks. The distinction is noted in [20] and [21]. Intermittently connected networks is properly a broader term for all networks where mobile nodes are able to route messages without end-to-end paths or fore-knowledge of the network topology. Likewise, delay tolerant networking is a broader term that includes space networks, the Interplanetary Internet, and other any network optimized for error-prone, constrained, delay-laden links. Delay and disruption tolerant networks are the intersection of the two: they route messages without end to end paths, are ad-hoc, and tolerate significant delays.

Note that in this thesis, as in the preponderance of the literature, DTN is used to describe delay and disruption tolerant networks (unless otherwise noted). Specifically, the class of networks focused on in this work are those for which the protocols discussed of the end of this chapter are designed, networks we term modern mobile DTNs. These are built around three central constructs: opportunistic forwarding, ad hoc network formation with autonomous node organization, and the ability to operate without reliance on infrastructure.

Opportunistic Forwarding

DTN routing solutions are necessary for challenged networks that violate one or more assumptions of the TCP/IP model: that an end to end path exists between the source and destination, round trip time is relatively small, and packet loss rate is low [6]. Thus a defining characteristic of modern DTNs is that the routing layer incorporates some mechanism that enables opportunistic forwarding when connections are available without certain knowledge of the destination. When nodes are mobile, this becomes opportunistic store-carry-forward, which enables routing between nodes that may never encounter one another [21]. Shen et. al. [22] and Ali et. al. [23] specifically discuss various strategies that rely primarily on mobility and store-carry-forward to overcome sparsely connected topologies. Opportunistic store-carry-forward is also a major component of the standard DTN architecture [6]. One exception to the DTN paradigm of store-carry-forward that is a burgeoning area of DTN research and bears mentioning is autonomous

sensor nets. The sensor fields are typically stationary, but use DTN routing mechanisms to enable a subset of the nodes to gather data from the other nodes opportunisticlly and transmit back to a control node or a central collection node when possible.

Opportunistic forwarding is complicated by a combination of factors common to DTNs: the mobility of the nodes, the geographic area over which the network is deployed, and physical layer factors. When mobility is high network topology is unstable and paths are short-lived. Further, the contact window during which nodes are in range of one another to exchange messages is short, making path prediction and the order in which messages are exchanged vital. Likewise, as less nodes are spread over a larger geographic area, it becomes less likely overall that nodes will meet. Finally, physical layer factors such as a small transmission range, radio interference, or jamming can significantly limit the ability of nodes to communicate.

Ad-Hoc and Autonomous Organization

As Dr. Fall describes in [7], the emphasis in DTN architectural development has steadily shifted from networks characterized by "scheduled connectivity in IPN case," to “opportunistic mobile ad hoc networks.” As also discussed above, opportunistic routing means that connections are leveraged if and when they are available. In the case of a sparse DTN, its possible that there will be many periods where no connections exist across the entire network; in this case, the entire network can be thought of as existing on an ad-hoc basis. When connections are able to be made, they need to be made without significant network configuration.

Nodes in a DTN are fundamentally stochastic, that is, they are required to make network and routing decisions at any given time based only on their current state–the information on hand. This is because nodes may also be disconnected for significant portions of time but still hold messages and need to participate in the network. Thus, protocols that require significant updating or configuration to participate disenfranchise nodes that aren’t able to form connections often. Thus, network configuration and dissemination of network characteristics can not be a requirement to participate. Routing information can be shared and used to gain knowledge of the network, which many protocols in fact do, but lack of that information can not bar participation in the network. Nodes, in effect, should be able to temporarily participate, on a ad-hoc, when available basis, in whatever routing mechanisms the protocol employs. This behavior also means that nodes are able to autonomously join and leave the network.

Many modern DTN routing protocols also attempt to logically organize nodes. Logical organization is often handled by the routing protocol and exploits network characteristics observable

by individual nodes. An example of logical organization would be to group nodes that are likely to see one another together and to route messages to any member of the destinations group. Such logical organization cannot be predetermined or controlled by a centralized, network wide mechanism, but again should be a autonomous, stochastic process that runs separately in each node.

Nonreliance on Infrastructure

Because modern DTNs are meant to overcome austere or challenged environments [6], they need to be able to operate without relying on supporting infrastructure. From the description of the ad-hoc behavior of DTNs, each node acts fundamentally as a peer-to-peer router. Therefore, DTNs can not be reliant on a framework outside of the nodes themselves to support network network operations. When infrastructure is available it can be leveraged to improve performance, but DTNs are designed to be infrastructureless.

Infrastructure does not only apply to actual external structures, but also applies to the nodes themselves. By the nature of DTNs there are bound to be periods where nodes are disconnected. Thus, the use of a central control or database node that connects to all nodes and organizes or coordinates routing behavior breaks the definition of a DTN (that would be more in line with a MANET). Therefore DTNs also can not rely on a single node, or even a subset of nodes, to coordinate routing network-wide.

2.2 Interest in DTN Routing

Having discussed how the vision for modern DTN evolved from early research into delay tolerant networks and described the characteristics of a modern DTN, it is appropriate to discuss how such networks have already been employed, why there is so much interest in the field, and what the future of DTN Networks may look like. Showing what this technology has already accomplished, why various sectors are interested in further research, and what the future of DTN's may look like is both the motivation behind and justification for this work.

2.2.1 Projects

There have been many recent examples of published projects that use DTN Technology technology to enable or further their research. These projects are discussed first to show how DTNs have already made significant contributions in other research areas. After presenting a few of the real-world DTN implementations that have already been fielded, we discuss interest in DTN's from the public and private sectors, and particular from the military. Finally, we believe

that DTNs are bound to play a increasingly prominent role in everyday life in the near future, and discuss some future applications of DTN technology.

Project Area One: ZebraNet, Canada Wildlife Monitoring, Environmental Deployment

Wildlife research is one field were DTN technology has already been used in several projects and has been proposed for use in ongoing and future projects, and DTNs are particularly well suited for tracking relatively scarce wildlife over large areas. One of the most widely-cited large-scale (geographically speaking) implementations of a DTN in was the ZebraNet project [10]. Also one of the earliest implementations of a DTN Network, ZebraNet actually predates the term DTN and research for this project Started in 2002 by Princeton Researchers attempting to track zebras at the Mpala Research Centre in Central Kenya. Collars that were capable of recording GPS locations and detailed activity logs were attached to 30 zebras. The collars tracked the zebra's movements and recorded them, along with physiological measurements and associated data. Each collar was also equipped with a short range radio, and acted as a short range peer-to-peer network node. The collars periodically scanned for other nodes in range, and exchanged data opportunistically when possible. Interestingly, several protocols were simulated and experimented with, likely making this the first DTN simulation research as well [10].

Traditionally wildlife and ecological monitoring has been done using spotters and helicopters or vehicles monitoring large areas, and statistically calculating the number of species in a given area [24]. With the success of ZebraNet, many other wildlife projects are now using similar techniques. One such project was recently proposed to track the movements of whitetail deer in Ontario, Canada. Under this proposal, a DTN sensor network would be deployed that tracks the number of whitetail deer observed over a sustained period of time. A series of mobile 'collector nodes' would be used to travel between the sensors periodically and collect the data. These researchers used simulation to show that the proposal would work, and contend that such a method remains the best option available to wildlife researchers [24].

Delay tolerant networks can also be used for environmental measuring studies such as the 2010 Postojna Cave project. During this project, a series of environmental monitoring devices were set up along a 20-mile cave in Slovenia. The devices collected data about the cave ecosystem and opportunistically transmitted that data whenever one of several tourist trains passed within range of the monitoring station [25].

Project Two: Diesel Net, Web Surf from a Bus

In 2004 the University of Massachusetts, Amherst, deployed a test bed for mobile networks called the Diverse Outdoor Mobile Environment, DOME. One of the major components of this project was placing 40 large peer-to-peer server/router combinations on transit buses. The buss routes centered on the campus, but also extended into the surrounding towns to cover over 150 square miles. Once established, researchers were able to access the framework and conduct experiments with a real-world implementation of data-mule network [26].

Using this test bed, a group of researchers at Amherst built a custom gateway and Internet browser that could operate over a DTN network. The buses were the mainstay of the network, though various other nodes also included over the life of the project; even turtles on the Amherst campus were used as mobile nodes at one point. The group implemented caching and DTN services on top of the peer to peer bus hardware, and demonstrated that web searches were possible over a scarcely populated large-range DTN network. While the average response time for a given Internet query was 2.7 minutes, far too long to satisfy an average Internet browser, the project shows the feasibility of developing real applications using DTN technology [27].

Rural and Remote Areas Deployment

One of the hopes for DTN Technology in the near future is to deploy such networks at the edges of the Internet to expand connectivity into scarcely populated and underdeveloped regions. This could be useful both in developing nations where traditional network connections are limited to major cities, and in developed nations as an alternative to metered connections, such as satellite, in rural areas.

One proposal to use DTNs to expand communications into remote areas is the Continuous Displacement Plans Oriented Network (CoDPON), which aims to provide remote medical services to rural commuities alogn the banks of the Amazon River. Proposed at ExtremeCom 2011, CoDPON is designed for deployment along the Amazon River in Brazil, where many of small and isolated communities are found along the banks of the river with no access to communications infrastructure or the Internet. This project aims to provide such villages with portable medical imaging devices, and use local boats that routinely travel up and down the river as DTN nodes to trnasmit the images to cities where doctors can review them.

This is one of many projects proposed at the annual Extreme Conference on Communication and Computing, ExtremeCom. For additional projects and information, see [28].

2.2.2 Current Interest: Public and Private Sphere

The world today is a networked world, but the networks today largely depend on expensive and complex infrastructure to operate. DTN technology can be leveraged to provide networked connectivity anywhere instantly, without regard to infrastructure or environment. DTNs also offer an alternative means of communication when traditional network links are incapacitated. This is largely why both public and private sector interest in DTN technology has been steadily growing in recent years.

Public sector interest has already been discussed, with environmental groups using DTNs to monitor wildlife [10], [24], and projects like Diesel Net extending public access information resources without paying for infrastructure [27]. Another obvious area in which DTNs could be leveraged to incredible effect is during disaster relief operations. Humanitarian assistance and disaster relief operations are often hastily organized, and allocating communications assets can be difficult on short notice. Traditional network infrastructure is often destroyed or overloaded, and responders require significant networking capabilities to communicate and coordinate. In Post Disaster Management Using Delay Tolerant Network, Saha et. al. propose using DTNs to facilitate communications between first responders after natural disasters, and simulate a disaster response scenario with all responder information system communications running over a DTNs [29]. Yutaka Sasaki and Yoshitaka Shibata alternatively propose using a gateway to the public broadcast systems and a DTN to disseminate information to the public following a disaster where the communications grid may have been disabled in Distributed Disaster Information System in DTN Based Mobile Communication Environment [30].

The private sector also shows signs of embracing DTNs, although private sector interest is slackened by the lack of a large-scale business model that incorporates DTNs. However, corporate research into vehicular DTN's that is likely to help connect the next generation of smart cars is under way, and it is likely that as the Internet of things continues to expand DTN technology will become essential to keeping those things connected. Apple has also included the APIs to implement MANETs and DTNs in newer versions of iOS called the Multipeer Connectivity Framework [31].

2.2.3 Current Interest: Military

The war fighters of today depend more on reliable access to information networks then ever before, and this trend is only likely to increase. At the same time, military operations tend to take place in challenging environments, where infrastructure or access to infrastructure is

limited and long-range communications can be easily jammed. The Department of Defense has invested tens of millions of dollars in developing readily-deployable DTNs to overcome these challenges.

In battle, a DTN could be used at all levels, from coordination between the squad and platoon level to coordination of the actions of dispersed and mobile forces across vast modern battlefields. The development of the DOD trusted hand-held mobile device is another reason to invest in DTNs that can connect front-line infantry hand-helds to artillery positions, mobile reinforcements, or orbiting medevacs and close air support units. Off the battlefield, the impact of DTN technology could enhance disaster response and relief efforts, improve connectivity during training exercises conducted in unimproved areas, and enable the tracking of convoys and supplies across an entire region. DTNs are also essential to the deployment of sensor nets and autonomous vehicles.

The Defense Advances Research Activity, DARPA, has been experimenting with various DTN technologies for years with the goal of creating a fully-connected battle space. The current DARPA Wireless Net After Next (WNAN) project incorporates DTN services that are shown to reduce bandwidth consumption over unreliable links by approximately 75% as compared to attempting to implement TCP over the same links [32].

2.2.4 Future Applications of the Technology

Internet retailers envision using tens of thousands of drones to deliver packages to consumers in the near future. Intelligent thermostats can already regulate household power consumption, and are starting to communicate directly with appliances to improve energy efficiency. Power and water meters are often read remotely from a passing truck. All of these networked devices are intermittently connected, and would likely benefit from DTN services. And the trend towards the Internet of Things is only increasing. The Internet of Things is an expression commonly used to indicate the current trend towards ubiquitous connectivity across all manner of devices. In the future envisioned under the Internet of things concept, wireless sensors and communicating automatons will be everywhere. Many of these devices are likely to be highly power-conscience, mobile, and intermittently connected. The only way to connect all of the low-power, short-range devices is likely to be through gateways and by forming disruption tolerant networks.

2.3 Routing Strategies

At a higher level of abstraction lies the underlying strategy employed by a protocol to address the difficulties of DTN routing. Many strategies have been proposed, some specific to a single protocol and others that describe the operation of several different protocols. Routing strategies are often customized to the physical attributes of a specific class of networks, and are highly dependent on certain node behaviors to support a particular approach to routing. For example, a routing strategy that relies heavily on node mobility like direct contact will obviously not function without mobile nodes, just as a strategy that relies on storage space will not perform well if nodes have small buffers. Common network attributes that most impact DTN routing performance are introduced below, and the underpinnings of routing strategy are discussed. We then present a brief survey of different routing strategies.

2.3.1 Network Attributes that Affect Routing Decisions

Node mobility and coverage are two attributes that greatly impact DTN routing protocol performance. Node mobility enables the store-carry-forward mechanism discussed earlier. It is through this mechanism that messages can be delivered to nodes that never encounter the originator of the message. Low mobility allows for longer encounters, which lets more messages transfer at each encounter, but decreases the likelihood of encounters between any two randomly selected nodes. High mobility increases the chance for encounters, but lessens the contact time during each encounter, reducing the number of messages that can be forwarded. Restricting node movement to a smaller area, such as forcing nodes to move only in a particular corridor, can also impact network performance. Grouping nodes together yields higher delivery ratio between those nodes, but makes connecting to nodes not in the restricted area difficult or impossible.

The aggegate node radio coverage of a network, that is, the area of the network divided by the area that a node's radio can rach, multiplied by the nubmer of nodes, also has an enormous impact on performance; high coverage results in more encounters and more routing options, but messages are more likely to fill the buffers of nodes causing memory management problems. Low coverage networks have to rely more on mobility to spread messages between nodes. Buffer space impacts protocol performance, as the ability to store many messages also allows greater dissemination of messages by more nodes. Generally speaking, more nodes carrying more typically results in higher delivery ratios.

Transmission speeds of node radios, environmental factors, obstructions, altitude, and any num-

ber of other physical characteristics can affect a routing protocol performance. Typically these attributes affect performance by altering the likelihood of encountering other nodes or altering the number of messages that can be exchanged during and encounter.

2.3.2 Replication

There are two fundamental approaches any routing protocol takes when exchanging messages between two nodes: single copy or multi copy. In the case of a single copy protocols, when messages are forwarded to another node the forwarding node deletes the message, leaving the receiving node as the only node carrying that specific message. Single copy is sometimes ambiguously referred to as "forwarding" [22], [33] when attempting to classify protocols into families by degree of replication. Multi-copy protocols forward many copies of a message to different nodes, with the goal of increasing the probability of delivery, but replicating messages causes increased overhead and resource consumption. Replicating protocols are likewise sometimes ambiguously referred to as the "flooding" family of protocols.

True single copy protocols are not typical in DTNs[1] as they tend to have a very low delivery ratio and higher latency then multi-copy protocols. However, Spyropoulos et. al. conducted a survey of single-copy DTN protocols in Efficient Routing in Intermittently Connected Mobile Networks: The Single-copy Case, finding, as expected, that single copy protocols do have much lower delivery ratios then their multi-copy peers, but are much more resource and power efficient [34]. Direct contact is the null-case of multi-copy protocols where the source node is not allowed to replicate or forward any messages, but can only deliver a message to its final destination. Direct Contact is occasionally simulated as a baseline comparison to evaluate mobility models and as a benchmark comparison protocol.

2.3.3 Knowledge

Routing strategies can also be categorized based on the amount of knowledge regarding the behavior of other nodes in the network is required by the routing protocol. Overhead is one measure of how effectively a protocol acquires knowledge of a network and uses that knowledge to make efficient routing decisions. The base case is where nodes gain no knowledge of network topology, and decisions are made only based on local knowledge and current connections. The infinite case would be a oracle protocol that has absolute knowledge of all current and future node behavior and is thus able to make perfect routing decisions [35]. Figure 2.1

[1] Single-copy DTN routing protocols are commonly used in situations where minimal energy consumption is the primary objective of the protocol.

shows a spectrum of contact schedule precision, and thus the knowledge that ca be gained of future contacts, inherent to different types of networks. Protocols can be designed such that nodes

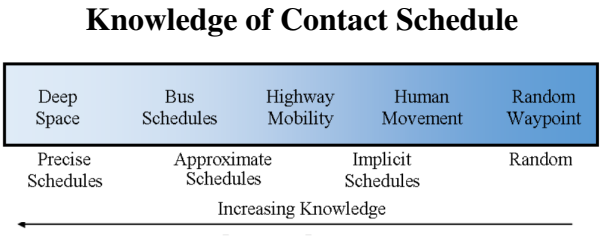


Figure 2.1: Knowledge Spectrum
Availability of Contact Schedule Knowledge based on Network Topology [35]

independently learn network topology information over time, or so that nodes collaboratively learn network topology by sharing information during encounters. In both cases nodes must remain stochastic; no central database or repository can be consulted and nodes have to rely on local information and information gained during encounters to build knowledge. It would violate the infrastructureless nature of DTNs to require that nodes consult a centralized knowledge collection agent.

2.4 Measures of Performance

A common set of benchmarks, or measures of performance, are required to compare different protocols. Other data points yield valuable insight into how a protocol is working or why critical measures of performance increase or decrease. Successful delivery of messages to their destination is typically the primary measure of performance, as DTNs are designed fundamentally to overcome intermittent connections. Latency, or the time that it takes a message to reach its destination, is also a measure of the performance of a DTN routing protocol but varies in importance depending on the applications serviced by the routing protocol. Other measures such as computational difficulty and energy efficiency, can also be important depending on the purpose of the network.

Other data points including the average buffer utilization–the ratio of memory used to store messages against the total memory, overhead ratio–the total ammount of data transmitted over the data used to transmit messages that reached their destination, and hop-count–the average number of nodes messages traverse before reaching the destination, provide valuable insight into how the protocol is working. Comparisons of these measures across multiple scenarios or sets of circumstances can help explain changes in performance metrics. Note that this work

does not implement a power model, but on average protocols that generate additional overhead or require more hops to route messages are less power efficient.

2.4.1 Delivery ratio and Latency

The delivery ratio, or the number of messages that reach the destination divided by the total number of messages generated, is usually the primary measure of DTN routing protocol performance. After all, routing protocol exist to deliver messages to the destination. Furthermore, if the given message does not reached the destination, any resources consumed in routing that message are wasted and add to overhead. Delivery ratio is tracked in every comparison and evaluation of protocols.

Latency, also called delivery delay [21], is the measure of the amount of time it takes for a message to reach its destination after it is generated. The relative importance of latency depends largely on the applications running above the routing protocol. Latency is important when evaluating network performance for applications where the user experience is tied to the wait time between messages, such as in the web search from a bus project from chapter 2.2.1 [27]. For products that catalog data over time, such as the wildlife tracking experiments described above, latency is less important. For further reading, Jones and Ward provide a comprehensive discussion of delivery ratio and latency in Routing Strategies for Delay-tolerant Networks [35].

2.4.2 Diagnostic Metrics

Several other measurements are helpful when inferring the behavior of a protocol from simulation statistics, or when assessing how different protocols operate. This work tracks three measures in addition to the performance metrics, including overhead ratio, buffer utilization, and hop count. Overhead ratio in particular is prevalent in survey literature, average hopcount is cometimes examined in experiments looking at energy consumption rates. Average buffer utilization can be used to infer the network load or frequency of contacts. Lower buffer utilization is also indicative of protocols that use knowledge to inform routing and buffer management strategy.

2.5 Protocols to compare

The baseline comparison protocols are typically Direct Contact and Epidemic Routing [9]. Direct contact can be considered the null case of the flooding family, where messages are only forwarded between source and destination nodes. Direct contact relies solely on node mobility to propagate messages; it incurs no overhead and requires no knowledge of the network.

Strict flooding is the opposite of direct contact, where all messages are replicated and forwarded to all nodes encountered. However, strict flooding only works when there is infinite buffer space and transmission speed. Since that is unrealistic, Epidemic Routing [9] is more often simulated. Epidemic is a modified flooding protocol. Messages are identified by a message ID, and at every encounter both nodes exchange a vector of all the message IDs in their buffer called the summary vector. Both nodes then exchange any messages whose IDs was not in the encountered node's summary vector. Acknowledgements are also flooded throughout the network and acknowledged messages are cleared from every node's buffer. Some implementations also include a Time To Live (TTL) field or other buffer management schemes that prioritize which messages are stored and which are discarded when buffers are full.

2.5.1 PRoPHET

The Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET) [36] is an extension of Epidemic routing, although it's more aptly described as belonging to the multi-copy, learning family of DTN protocols strategies. In PRoPHET each node encountered is assigned a initial delivery predictability value (DP) based on a initialization constant, P_i, between zero and one. Whenever a new node is encountered it is added to the summary vector and becomes a known node. When known nodes are encountered, their DP is increased according to Equation 2.2. DP is also lowered over time according to Equation 2.1, where γ is a aging constant between zero and one, and Δt is a time-step constant. Nodes are also aware of transitivity, that is, if they have not seen another node directly, but they see an intermediary to that node often enough that it is likely a message could reach the unencountered node through the intermediary. Transitivity DP to reach node c via node b for source node a according to Equation 2.3, where β is the transitivity scaling constant, and is included in the summary vector as well.

$$P_{a,b} = P_{(a,b)_o ld} + (1 - P_{(a,b)_o ld}) * P_i \tag{2.1}$$

$$P_{a,b} = P_{(a,b)_o ld} * \gamma^{\Delta t} \tag{2.2}$$

$$P_{a,c} = P_{(a,c)_o ld} + (1 - P_{(a,b)_o ld}) * P_{(a,b)} * P_{b,c} * \beta \tag{2.3}$$

At each encounter nodes exchange their summary vector containing message IDs for all messages stored in the buffer and the DP vector described above. Messages are only exchanged if the encountered node has a higher DP value for the destination of the message then the current node. The original version of PRoPHET occasionally encounters a situation called the parking

lot problem, where a group of nodes oscillate their summary vectors, which results in very high DP for a few nodes and very low DP for all other nodes. To solve this problem PRoPHETv2 [37] uses an encounter timing mechanism that scales the degree to which a encounter can affect DP based on how recently the node was last encountered. PRoPHETv2 is commonly used in place of the original PRoPHET protocol to avoid parking lot situations [36].

2.5.2 MaxProp

The MaxProp [38] protocol is also an extension of Epidemic routing and was originally designed for use in vehicular-based DTNs, but is widely used due to its excellent performance across a wide range of scenarios. MaxProp forms a summary vector similar to PHoPHET that contains message IDs and a routing metric called Delivery Likelihood (DL). At every encounter messages that are addressed to the encountered node are exchanged first, followed by the summary vector and a vector of acknowledged messages.

Maxprop then sets aside a portion of the encounter to transfer messages with a hop count below threshold value t. To determine t, MaxProp tracks the average average number of bytes transferred per encounter, x, the buffer size, b, the the portion of x, p, that should be used to transfer low hop count messages. p is determined by Equation 2.4, and t is set such that when messages are ordered by hop count, the hop count of the last message in p equals t.

$$\begin{aligned} &\text{If } x < \frac{b}{2}, \text{ then } p = x \\ &\text{If } \frac{b}{2} \leq x < b, \text{ then } p = \min(x, b-x) \\ &\text{If } b < x, \text{ then } p = 0 \end{aligned} \tag{2.4}$$

Finally, the remaining messages are forwarded ordered by DL, with the messages most likely to be delivered forwarded first. To determine DL MaxProp builds a directed graph that includes all nodes in the network, s, and each edge is assigned a DL equal to $\frac{1}{|s|-1}$. When node a encounters node b, the DL_b^a is incremented by one, then all DL's are normalized to 1. Over time, nodes encountered more frequently obtain larger values. When the summary vector that contains b's DLs is received and node reachable through b with a higher DL are added to a's graph. Dijkstra's algorithm is then used to compute path costs from a, where for each DL assigned to a edge on the path, $Cost = (1 - DL)$. MaxProp drops messages with low DL when they are forwarded to a node that has high DL to the message's destination when additional buffer

space is required. MaxProp also drops messages that have high hopcount when buffer space is required, as it is most likely that messages with the highest hop count have been delivered by another node [38].

2.5.3 Vector

The Vector Routing protocol [1] is a selective flooding protocol that uses the difference in the direction of movement between a node, a, and a encountered node, b, to determine the number of messages to replicate and forward, $nFwd$. To accomplish this each node is required to keep a list of its most recent N positions updated every ΔT seconds. Upon encountering a node, both a and b calculate and exchange their current direction, θ, which is then used to determine $nFwd$. More messages are transferred to nodes traveling in orthogonal directions, as those nodes are more likely to encounter nodes different nodes then the original, and fewer messages are transferred to nodes traveling in parallel directions as they are more likely to encounter the same nodes as the originator.

Vector extends Epidemic to form a selective flooding protocol in order to reduce overhead and latency while retaining similar delivery ratio to epidemic. When the vector mechanism is used with a standard flooding protocol such as Epidemic and stimulated in a random mobility scenario, researchers noted little change in the delivery ratio but 28-38 percent reductions in latency and overhead as compared to pure epidemic routing [1].

2.5.4 GAPR

GAPR [2] extends MaxProp and uses the MaxProp *DL* mechanism at each encounter to assign messages a probability, P, that the message will be delivered if forwarded to the encountered node. GAPR also leverages geographic data to identify and reset invalid *DL*s. To accomplish this, GAPR also maintains a location list of where and when nodes were last encountered.

When encounters occur, nodes first exchange vectors of acknowledgements and clear buffers of acknowledged messages. Any messages destined to the encountered node are then forwarded. Following the exchange of directly delivered messages, nodes exchange probability data and location lists. Each node compares the location list and resets the P value of any node identified as having made a unusual change in position in a short ammount of time. The local location list and the recieved location list are then merged, and all remaining messages are ordered by P value and forwarded in order [2].

2.6 Literature Review

Significant work has already been completed and published in the area of DTN evaluation and simulation. Reviewing the substantive literature from this field enhances the quality of this work and is required to show that the significant body of knowledge developed across the spectrum of related research is employed, and to show that this work both builds on and expands that body of knowledge. To that end, several survey papers and their conclusions are examined below, followed by discussion of the key takeaways from other papers that have simulated new protocols, and how those protocols presented simulation data.

Survey papers reviewed for this work include [20], [21], [23], [33], [35], [39], and [40]. While some of these papers predominantly summarize different theories and strategies behind routing decisions, others are grounded in real-world protocols. In particular, Cao presents a comprehensive chart of nearly 100 protocols and each one's replication behavior, bandwidth, buffer use, and energy [21]. Most survey papers do not use origional simulation to make comparisons; Lo et. al. do in Routing and buffering strategies in delay-tolerant networks: Survey and evaluation [39], but the scenario settings are not included and delivery ratios are consistently under 20 percent using ONE. This appears abnormally low compared as compared to other simulation work and seems to indicate a very austere environment was simulated or there were simulation errors, possibly causing abnormal comparison data.

Every survey and comparison paper made a distinction between single-copy and multi-copy protocols, though Direct Contact tended to be the only single-copy protocol included. Lo et. al. [39] determines that high-replication protocols of outperform protocols with lower replication in every case, even with constrained buffer size. In Routing Strategies for Delay-Tolerant Networks [35], Jones determines that hybrid protocols, which employ knowledge to vary the degree of replication on a learned by-encounter basis, are the best option for DTN routing. Other comparison papers stop short of drawing a broad conclusion on the suitability of a certain family of protocols over a broad set of circumstances. In A Survey Paper on Routing in Delay-Tolerant Networks, Puri et. al. [40], concludes only that most protocols are designed to perform well only under a certain set of conditions, while Mangrulkar et. al. [33] goes further yet, stating that a protocol suitability must be based in the application for which it was designed, referring to the network characteristics. Ali sees the least potential for cross-purpose use of protocols under different scenario circumstances: "there is not [a] routing protocol that would cater [to] different environments ... Every routing protocol can only work for one set of environment

conditions and would drastically fail in different network situations" [23].

Notably, a point raised by Cao is the difficulty many proposed learning protocols, or more specifically, protocols that use historic information to make routing predictions, encounter is how to differentiate between useful information and superfluous or, worse, inaccurate and detrimental information [21]. This is in line with one of the objectives of this work, to combine Vector and GAPR in order to leverage Vector's replication limitation mechanism with GAPR's ability to intelligently replicate and discard poor-quality information. Vector, and thus GAPR2, is designed to further control replication based only on real-time local information, not shared information that could become outdated or inaccurate. Meanwhile, GAPR2 uses learned information twice, both to rank message exchanges and to validate those ranks. This satisfies the use case recommendations of most of the survey authors, as vector only applies controls on replication if short-term condition, namely that nodes are traveling in parallel, is met, while GAPR leverages additional knowledge of network topology to order forwarded messages.

Simulation and evaluation papers reviewed for this work include [1], [2], [9], [36], amd [38], which includes each protocol discussed in the background and to be used as comparison protocols during analysis. Also reviewed were three publications that specifically address evaluating DTN protocols, [8], [39], and [41]. Evaluation through simulation is the norm when working with DTN routing protocols because DTN evaluation is trending toward complex large-scale mobility models that require simulation to adequately evaluate, as mathematical descriptions of such complex systems using Markov models and differential equations are too simplistic to realistically represent the behavior of such systems [39].

Simulation of mobility can be done in one of three ways: through a historical trace of the actual movement of real-world nodes in a system as done by Song in Evaluating opportunistic routing protocols with large realistic contact traces [41], through a randomized mobility generator, or through a scenario based partially on realistic characterization and partially on randomization as Grasic [8] and Lo [39] favor. When using either of the latter two, Grasic, [8], notes that constraints imposed by the context of the simulated scenario should be maintained across simulation trials. For example, if modeling cars on a highway, it would be counter productive to allow mobility speeds exceeding reasonable top highway speeds, and if modeling sensor nodes with 1GB of memory, altering buffer space across trials is not necessary. However, routing performance is highly dependent on node mobility, density, and distance between sourc4.10c show the effect of increasing transmissions speed on latency. In both high network traffic cases the

latency of GAPR2 and Vector become asymptotice and destination [39]. Therefore, simulation should range across not just several mobility models, but ranges of values that are rational for the given scenario.

Publications that simulate protocols present various measure of performance is differing ways. All papers presenting routing performance data used at a minimum several line graphs. Balasubramanian et. al. in introducing Epidemic Routing [9] present independent variables, namely performance metrics, as series of lines on a single graph, each axis of the graph representing a dependent variable. Each run altered two dependent variables, repeated for several sets of independent variables. These included radio range against delivery latency against percent of messages delivered, message hop limit against delivery latency against percent of messages delivered, and buffer capacity against latency against delivery ratio. This was only possible because [9] only presented data on the Epidemic protocol, vice later simulation papers that tend to present multiple protocols for comparison.

Vector [1] was designed to reduce the overhead, average hop count, and latency of epidemic routing, thus the presentation of data included line graphs of both protocols that shows delivery ratio, overhead, and hop-counts the number of nodes is varied. MaxProp [38] presents line graphs representing an aggregate summary of 20 simulation runs, each run using different mobility and node density settings. Two graphs present delivery ratio and median latency against as message generation rate increases, and two graphs present delivery ratio and median latency as buffer size increases. GAPR [2] uses a combination of bar graphs and line graphs to show the results of two mobility scenarios. Line graphs show delivery ratio as buffer size and transmit speed are varied, and bar graphs show average latency and overhead ratio against buffer size and transmit speed. Finally, PRoPHET [36] uses line graphs to show message generation against delivery ratio, latency, and overhead.

Every protocol simulation paper presents, at a minimum, findings on delivery ratio and latency, and some include overhead ratio and hop count as well. Dependent variables graphed across the range of simulation papers in order of frequency are buffer size, message generation rates, transmit speed, number of nodes, radio range, and message hop limit.

CHAPTER 3:
Methodology

In this chapter, we present and justify the specific tools, methods, and approaches used to complete our objectives. This begins with introducing and defending our choice to use The ONE Simulator and how Vector and GAPR2 are implemented in Java for the ONE environment. We then describe each scenario, the scenario being the construct that informs and justifies simulations. Finally, we outline each set of simulations and the settings of each simulation set, explaining why we choose to present and analyze specific dependent and independent variable combinations within each set. We also explain the process of generating a custom simulation scenario in Section 3.7.

3.1 Simulator

The simulation engine chosen to work with is central to any simulation-based experiment, as every trial will be enabled and constrained by the simulation engine. Any simulator considered must be capable of supporting the scenarios included in the experiment accurately, but also should not include unneeded complexity that will hamper large-scale simulations. The simulator needs to support the extension of existing protocols and support the creation of new protocols. It should also be well-documented in order to facilitate modification to specific simulations, and finally should support common file extensions for other inputs.

There are two well-known simulators used widely in DTN research, the Network Simulator 2 (NS-2) and The Opportunistic Network Environment simulator (ONE). NS-2 is an event driven simulator, developed through wide collaboration between several colleges and research firms, and models link layer through application layer network behavior. NS-2 is a open source project and includes a variety of user-developed extensions, protocols, and customizations [42]. The ONE Simulator is also an event based simulator that was developed at the Helsinki University of Technology specifically for simulating DTN routing protocols [3]. The most complete source of information on ONE is the project web page, [43], which includes the simulator source code, complete Javadoc's for all libraries, and links to several tutorials and resources.

Grasic et. al. in A Survey Paper on Routing in Delay-Tolerant Networks [8], find that roughly a third of researchers use NS-2 and a third of researchers use ONE to conduct DTN simulation across widely cited published work, with the remainder conducting simulations in custom-

implemented simulators. Further, Grasic finds that when custom simulators are used, the researchers almost never discuss the validity or availability of the coding. This leads to difficulty in verifying and recreating past experimentation–a cornerstone of properly conducted scientific work. This is evidenced by the experience of Grasic et. al. in attempting to investigate a particularly poor-performing simulation of ProPHET described in [44]:

> After [the] lengthy process of retrieving the custom simulator source code from authors (with their full cooperation), close examination of the PRoPHET source code was conducted. Despite the vast effort put in the process of implementing the simulator and other routing protocols, two bugs were found in the code that completely hindered the PRoPHET routing scheme.

Thus, in keeping with the best practices described by Grasic et. al. [8], and to facilitate recreation of the results obtained by this experiment a open source simulator, NS-2 or ONE, is used.

NS-2 is a multi-purpose network simulator that includes models for all layers, physical through application, and can be used to model many classes of networks, from traditional wired networks to MANETs and DTNs. NS-2 is particularly popular in MANET research and DTN research looking at other layers of the DTN architecture such as bundling protocol performance. The aim of this experiment, however, is to isolate and simulate only the routing protocols. Layered network models are neither implemented nor necessary for the simulations in this thesis, and would add both unnecessary complexity and additional variables that would require further controls. A final benefit to using ONE is that the original GAPR work is conducted in ONE, so using ONE for this work allows for reuse of the GAPR source code with relatively minor updates to account for version changes.

3.1.1 ONE Architecture

ONE is coded in Java, with a series of Java packages (libraries) that are coupled and dependent on each other to run simulations. The relationship between these packages is shown in Figure 3.1, as specified in [3]. The packages that most directly affect this work include include the core, movement, map movement, and routing packages.[2]

Each simulation is controlled by a configuration file which provides settings to each package. Simulations are run in batches, or sets of dependent variables, as defined in the configuration

[2] All packages were used and are required to run simulations in ONE. The packages highlighted here are the packages that were altered to support specific simulations or vital to our understanding of the ONE environment.

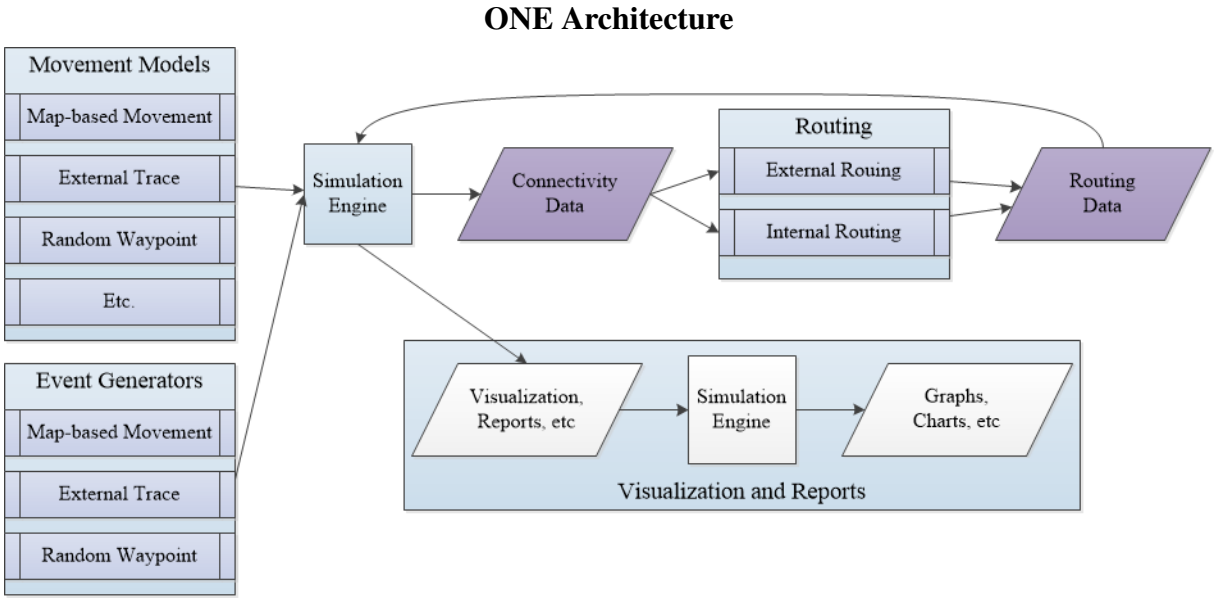


Figure 3.1: The ONE Simulator Architecture Flowchart

file. The path to the configuration file is given as a command line argument to the ONE startup script, one.sh, along with the batch size (number of simulations to run in a set) and other options. In this work, large batches of simulations were run from the command line but specifying batch mode with the -b option followed by the number of simulations and the configuration file: *./one.sh -b [number of simulations] [configuration file]*. If the batch mode option is omitted the GUI is loaded, which is particularly useful when designing custom mobility maps and to validate that the first run of a batch is proceeding as expected.

The core package contains the simulator's main class, DTNsim, which is responsible for starting and simulation, parsing batch mode operations, reading the command line arguments, and calling the constructors or controllers for each of the other packages. The other classes in the core package manage basic simulator function such as the SimClock class that controls simulated time and the Connection class that provides the basic structure for node encounters.

The movement package contains classes for the various mobility generators employed during simulations. In this work several mobility generators are called including RandomWalk, MapBasedMovement, MapRouteMovement, and ShortestPathMapBasedMovement; each of these is discussed further in the mobility section below. A subsection of the movement package controls map based movement (Movement.Map), which is capable of reading files in well-known text (WKT) format and constraining node movements to paths defined by the WKT file. WKT

is a format that describes geometry by lists of points (ONE actually reads these as map nodes, but in the interest of simplicity and not overloading the term node, we refer to them as points). When a map-based mobility generator is used, nodes are only able to travel between congruent WTK points. How WKT files can be generated to form a custom scenario is explained in more detail in Chapter 3.

All routing protocols are contained as classes within the routing package, each as a separate file. The abstract superclass MessageRouter is the parent of all routing protocols, and controls the most basic routing behavior common to all protocols. This is discussed further in the following section. ONE also includes a graphical user interface (GUI) that allows users to visualize the progress of simulations as seen in Figure 3.2. Each blue icon represents a individual node, and the green circle surrounding each node indicates the node's radio range. A black line between nodes is a visual indication that those nodes that have established a connection. Selecting a specific node from a list on the right highlights the selected node's current path and shows detailed statistics for that node. At the bottom of the GUI, a running list of all encounters and message transfers is displayed.

The ONE Simulator Graphical User Interface

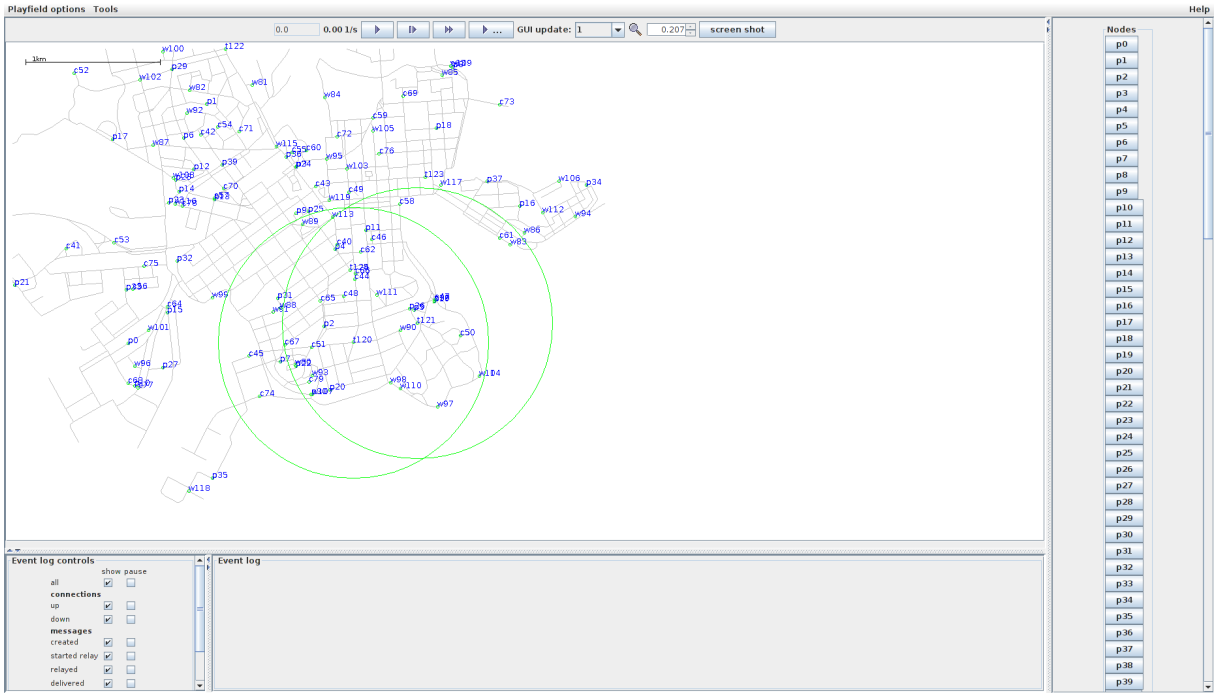


Figure 3.2: ONE GUI and Helsinki Map

3.2 Protocol Implementation

The concepts on which Vector routing is based were first proposed by Kang and Kim in Vector Routing for Delay Tolerant Networks [1], but no implementation details or code was included in the publication. The protocol coded for use in ONE leverages the concepts found in [1], but the implementation is original, and thus it is essential to document how our vector protocol works. We also implement GAPR2, a modified GAPR protocol that combines the Vector and GAPR protocols. The implementations of both protocols are described below, including the major logical functions and parent classes.

3.2.1 Vector

Vector was coded as a extension of a ONE abstract parent class called ActiveRouter, which is in turn a extension of the abstract superclass MessageRouter. MessageRouter is the superclass for all routing protocols in ONE, and supports routing functions required for all routing protocols to work such as initializing a buffer and tracking active connections to other nodes. MessageRouter also handles most of the basic interfacing with other packages such as simulation world updates and maintaining routing statistics for the reports package.

ActiveRouter provides useful methods for routers where nodes actively manage connections to other nodes, message transfers, and buffer space. Our implementation of Vector rides on top of ActiveRouter, and thus inherits the ActiveRouter connection management and buffer management functions. This means that our baseline Vector protocol has some of the buffer-space reduction mechanisms found in MaxProp, GAPR, PRoPHET, and other active protocols. Namely, VectorRouter will drop the oldest messages[3] in its buffer to make room to new messages in accordance with ActiveRouter's buffer management functions. However, messages in the buffer are unordered and no specific logic is added to the baseline Vector protocol to enhance knowledge beyond that described in this section.

To implement the vector logic, each node records its location every second and retains the most recent ten locations in an arrayList. When two nodes form a new connection, each node calculates a weighted average of the distance traveled in the X and Y directions, then uses trigonom-

[3]Here we mean the oldest message received, not by TTL

etry to calculate current heading, Θ, normalized to the positive X axis according Equation 3.1.

$$
\begin{aligned}
&\text{If } X \geq 0 \text{ and } Y \geq 0\text{:} \\
&\qquad \text{If } X = 0\text{, Then } \theta = \frac{\pi}{/2} \\
&\qquad \text{Else } \theta = tan^{-1}(\frac{Y}{X}) \\
&\text{If } X < 0 \text{ and } Y \geq 0\text{:} \\
&\qquad \text{If } Y = 0\text{, Then } \theta = \pi \\
&\qquad \text{Else } \theta = tan^{-1}(\frac{|X|}{Y}) + \frac{\pi}{2} \\
&\text{If } X < 0 \text{ and } Y < 0\text{:} \\
&\qquad \text{If } Y = 0\text{, Then } \theta = \frac{3\pi}{2} \\
&\qquad \text{Else } \theta = tan^{-1}(\frac{X}{Y}) + \pi \\
&\text{If } X \geq 0 \text{ and } Y < 0\text{:} \\
&\qquad \text{If } X = 0\text{, Then } \theta = 0 \\
&\qquad \text{Else } \theta = tan^{-1}(\frac{Y}{|Y|}) + \frac{3\pi}{2}
\end{aligned}
\tag{3.1}
$$

Nodes first exchange a list of acknowledged messages and clear buffers of any messages found in the list of acknowledgements, then exchange their current heading. Each node calculates the difference between the two headings according to Equation 3.2, where the local heading is θ, the received heading is Θ, and the difference is $\delta\Theta$:

$$
\begin{aligned}
&\text{smallAngle} = min(\theta, \Theta) \\
&\text{largeAngle} = max(\theta, \Theta) \\
&\text{ccwAngle} = 2\pi - \text{largeAngle} \\
&\delta\Theta = 2\pi - (\text{smallAngle} + \text{ccwAngle}
\end{aligned}
\tag{3.2}
$$

Finally, vector attempts to replicate and forward a portion of buffered messages, *msgLimit*,

according to 3.3 where $\delta\Theta$ is determined by Equation 3.2:

$$\begin{aligned}
&\text{if } \delta\Theta < \pi/12 \text{ or } \delta\Theta > 11\pi/12, \delta\Theta = 0 \\
&\text{if } \delta\Theta < \pi/6 \text{ or } \delta\Theta > 5\pi/6, \delta\Theta = 0 \\
&\text{if } \delta\Theta < \pi/4 \text{ or } \delta\Theta > 3\pi/4, \delta\Theta = 0 \\
&\text{if } \delta\Theta \geq \pi/3 \text{ or } \delta\Theta \leq 2\pi/3, \delta\Theta = 0
\end{aligned} \tag{3.3}$$

We found Vector to be a interesting and novel approach to DTN routing; several improvements beyond the scope of this work are possible with Vector and are discussed in chapter five.

3.2.2 GAPR2

The GAPR protocol presupposes that a node has access to a geolocation tool. We thus reason that a node capable of running GAPR would also be capable of running Vector, as the only requirement for Vector is that a node is able to access a service that can supply locations. Further, GAPR tends to generate significant overhead. Thus, we combine the logic of Vector and GAPR in an attempt to lower GAPRs overhead without significantly impacting delivery ratio. The delivery probability and threshold algorithms described in 2.5.2 are used to arrange messages and prioritize delivery. In GAPR2, however, when nodes first encounter they also calculate and exchange their current vector using Equation 3.1. Before messages are ordered as described in Section 2.5, each node calculates its angle of incidence to the other by Equation 3.2. Once messages are ordered, Equation 3.3 is computed and the returned value limites the maximum number of messages to be forwarded, in the order determined by GAPR logic.

3.3 Scenarios

A scenario is the framework around which a simulation is built–a description of what is being modeled by each simulation. This experiment modeled three scenarios: the Helsinki City Scenario, the Random Mobility Scenario, and the Military Evacuation Scenario. The following three sections describe each scenario, the reason the scenario was simulated, and the mobility generator employed.

3.3.1 Helsinki Scenario

The Helsinki Scenario included with ONE is likely the most widely-used simulation in DTN research. It is built using a series of WKT files, each representing a separate map layer of the city of Helsinki, Finland. The base layer is a depiction of all main roads throughout the city. Another

layer indicates all pedestrian walkways and open areas in the city, and yet another describes the tram tracks through the city. A final layer has the location of points of interest (POI) like shops, parks, and offices throughout the city. Each WKT file corresponds to a particular group of nodes–cars to roads, pedestrians to walkways, and trans to tram tracks. Each group of nodes is only allowed to travel on the routes provided on their relevant WKT files.

There are two mobility models that were built for the Helsinki Scenario that, while not used in this work, are worth mentioning to highlight the degree to which mobility models can accurately depict real-world mobility. These are the Working Day and Helsinki Nightlife models. The working day model replicates daily workday life in Helsinki, modeling nodes starting the day in residences, communing to work, and moving around a office building. Some nodes go to lunch and some nodes are tourists, traveling between shops and other POIs. Working nodes return to residences at the end of the workday. The nightlife scenario is similar, with nodes traveling to attractions and POIs open at night. The realism of both models is remarkable, so much so that they were verified against real-word traffic data and traces [45].

This work is not concerned with such an exacting degree of realism, but rather with generating comparison data that shows the performance of various protocols when they are restricted to a city-like environment and that can be recreated or referenced against other works. Further, the working day and nightlife scenarios severely limit the movement of most nodes once they reach their destination (workplace or entertainment venue). Any map-based mobility generator can be used to govern the mobility of the nodes in the Helsinki Scenario.

Therefore, the shortest path map-based mobility generator is used for pedestrians and automobiles, and map route movement is used for the trams. Cars and Pedestrians select one of the POIs and travel to it using the routes determined by their WKT map file. Once reaching the POI, the node pauses for up to two minutes, then selects a new POI and begins traveling to it. The trams follow the route from the tram WKT file, pausing for between ten and thirty seconds at each point. This retains the realism of the Helsinki City infrastructure, but provides consistent movement over the length of the scenario. This is also the more commonly implemented mobility generator for the Helsinki Scenario, which allows the results of this simulation to be compared to a wider range of results from other work.

Under map based movement node destinations are controlled by a random number generator which is used to select a POI (representing various shops and attractions) from the POI WKT file. Nodes then travel to the POI as described above. Nodes are assigned attributes by groups,

with six groups of nodes being the default Helsinki setting. Two groups of 40 nodes are used to represent pedestrians, one group of 40 nodes represents cars, and the final three groups represent trams (with 2 cars per tram). Each node group is assigned speeds appropriate to the what they are simulating. Pedestrian and car radios have a range of ten meters, and trams have a range of one thousand meters. The map of Helsinki with nodes and respective radio ranges can be observed as the map overlay used in the image of the ONE GUI, Figure 3.2, in the first section of this chapter.

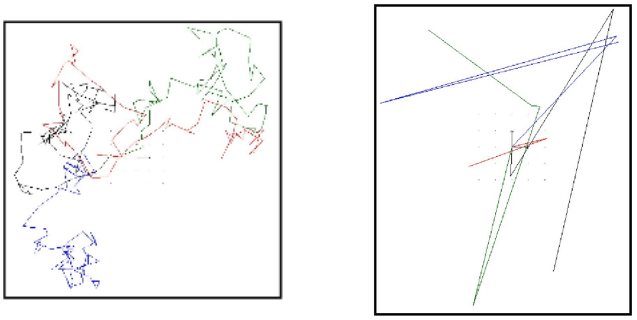
3.3.2 Random Mobility Scenario

There are two common random mobility generators: random waypoint and random walk. Random waypoint generates nodes at a random position within the scenario grid. Each node then travels from their origin point to a randomly selected destination point on the grid. Once they reach the destination, they simply select a new random destination and travel there. The problem with random waypoint is that nodes tend to be concentrated in the center of the scenario grid, vice evenly distributed over the entire grid.

The random walk generator also creates nodes at random locations on the simulation grid, but nodes select a random direction to travel in and a random amount of time to travel in that direction.[4] If a node attempts to travel past an edge of the grid, they are reflected back onto the grid in the opposite direction. The result, as shown by Broyles et. al. in [46], is that nodes tend to spend a more equal amount of time between the center and near the edges of the grid. Figure 3.3 shows a trace of the random walk and random waypoint generators from [46]. This work employs a random walker mobility generator to more uniformly distributed mobile nodes across the simulation grid.

The objective of the random mobility simulations is to show the performance of the various protocols in a situation where knowledge of the network topology is difficult gain and predict, and to determine how speed and network size effect the performance of the protocols. The primary factor altered between runs to influence node mobility is node speed. Other factors, such as the duration that nodes pause at their destinations, also impact random mobility but are maintained as constants (controls) across these simulations.

[4]Note: the random walk mobility generator implemented in ONE uses distance in meters vice time to determine how long nodes travel in a given direction.



(a) Random Walk (b) Random Waypoint

Figure 3.3: Mobility Generator Traces in NS-2

3.3.3 Military Scenario

Bold Alligator is a annual multinational military exercise designed to "revitalize, refine, and strengthen core amphibious competencies, which are critical to maritime power projection and are a cost-effective option for a wide range of military operations" [5]. In 2012 the live exercise was the largest navy-marine corps beach landing exercise in over a decade. The exercise is conducted around Camp Lejeune, North Carolina. The exercise is built around a scenario wherein a rogue country named Garnet attempts to invade a country called Amberland, which precipitates a build-up of U.S. forces to support Amberland and an evacuation of U.S. citizens [5].

The Bold Alligator planning material and precepts are used as the basis to construct a scenario, but this scenario is not an attempt to historically recreate the exercise.[5] Instead, a realistic scenario is built that models what could happen during the evacuation portion of the exercise. The result is complex scenario that uses over 400 nodes to simulate the various elements of a Marine Corps Expeditionary Unit (MEU) as the MEU conducts a large-scale security and evacuation operation of the 50 square kilometer area area surrounding Camp Lejeune. The scenario is described below:

Amberland is a United States partner nation that hosts several thousand U.S. civilians, mainly concentrated around a joint American-Amberlantean base named Camp Lejeune. For the past several months relations between the U.S. supported nation of Amberland and its northern neighbor, Garnet, have rapidly deteriorated. In the past few days there have been isolated skirmishes along Amberland's northern border border skirmishes, and a U.S. Marine Amphibious

[5]This thesis is based only on readily available open-source information. Due in part to this and also to time constraints, we chose not to attempt an exact replication of the Bold Alligator exercise, but instead a fictitious scenario based on the guidelines of the Bold alligator exercise.

Readiness Group (ARG) with an embarked Marine Expeditionary Unit (MEU) was deployed to the coast of Amberland. Overnight the border skirmishes have turned into a full-scale Garnetian invasion of Amberland. The Government of Amberland has requested assistance, and the MEU has been directed to secure the area around Camp Lejeune and evacuate the civilians from the surrounding towns. The scenario is depicted below in Figure 3.4. U.S. forces are to take positions within the area outlines in green and brown southwest of the two impassible swampy, forested areas shown in green. Garnetian forces are concentrated northeast of the U.S. deployment area and are invading southward as illustrated by the red box and arrows.

Bold Alligator Beach Landing

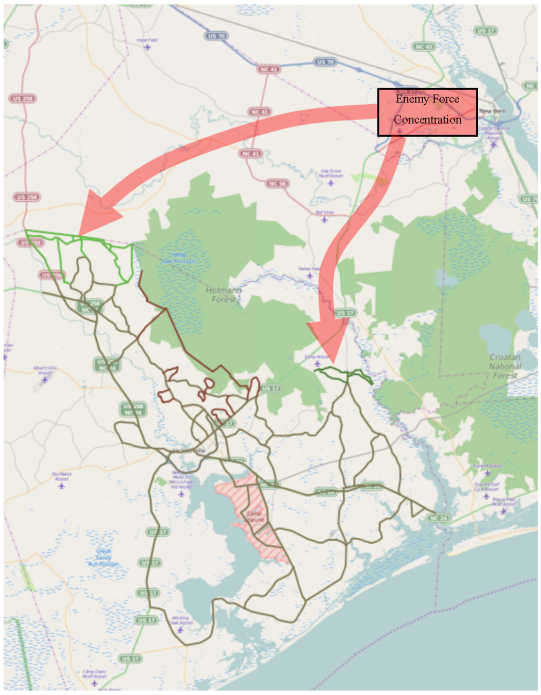


Figure 3.4: Overview of Military Scenario

Garnetian forces have destroyed the main Amberlantean telecommunications infrastructure during targeted strikes against the Amberland command and control systems and are intermittently jamming satellite communications. Fortunately, this MEU is equipped with a new generation of DOD trusted hand-held PDAs that can interface with one another to form a DTN. Further, heavy equipment was retrofitted to include a DTN node that supports storing and forwarding messages. Each platoon is issued 50 hand-helds to facilitate communication. Platoons, Humvees, and the

ARG generate messages to coordinate the dispersed forces; the marines in the platoons generate a large number of small messages, where the Humvees and ships generate large messages less frequently. The attributes of each DTN node are summarized in Table 3.1.

Marine Expeditionary Unit				
grps / size	Name	Speed (Km/h)	Radio Range / rate	Disbursement
7 / 50	Marines	4 - 10	100m / 125kBps	One grp per area
1 / 20	Humvee	50 – 80	3 km / 250kBps	Transits between platoons
1 / 2	Drone	40 - 70	4 km / 125kBps	Drone Surveillance route
4 / 2	Helicopter	150 – 250	5 km / 500kBps	Transit, ARG to platoons
1 / 2	LCAC	450 – 600	6 km / 500kBps	Transit, ARG to shore
1 / 1	ESG	5-15	10 km / 2.5MBps	MEU steaming box

Table 3.1: Military Node Attributes

The ground combat element in this scenario is deployed as 7 platoons and each platoon has 50 hand-helds that interface with the DTN network. Each platoon is deployed to a specific area within the U.S. deployment zone shown in Figure 3.4 and patrols one of the smaller areas outlined in different colors in Figure 3.5. The movement of the marines in each platoon is controlled by a map based mobility generator, which models the marines moving randomly around the roads and neighborhoods in their assigned area. 20 Humvees with vehicle-mounted DTN nodes and transmitters travel around the main roads (shown in Figure 3.4) to patrol the area and supervise the evacuation progress.

Two drones are assigned to patrol the northern front of the area for surveillance, and follow the dotted pink line in figure 3.5. The drones have a DTN node that has a large buffer, but low transmission speed and range. Four of the platoon patrol areas are also evacuation points, and teams of two helicopters transit between the evacuation points and the ARG carrying evacuees. The route the helicopters follow is shown in dotted red lines on the map. Helicopters carry the same DTN node as the Humvees, and pause at each end of their route for 30 minutes (to fuel on the ships, and to load evacuees at the landing zones). Two LCACs are assigned to facilitate evacuations in the platoon deployment area closest to the ARG, shown by the yellow dotted line on the map, and the ARG is represented by three nodes patrolling 5 miles offshore. The LCAC also carries the same node that the Humvees do, and the ships have very powerful, long range DTN nodes on-board.

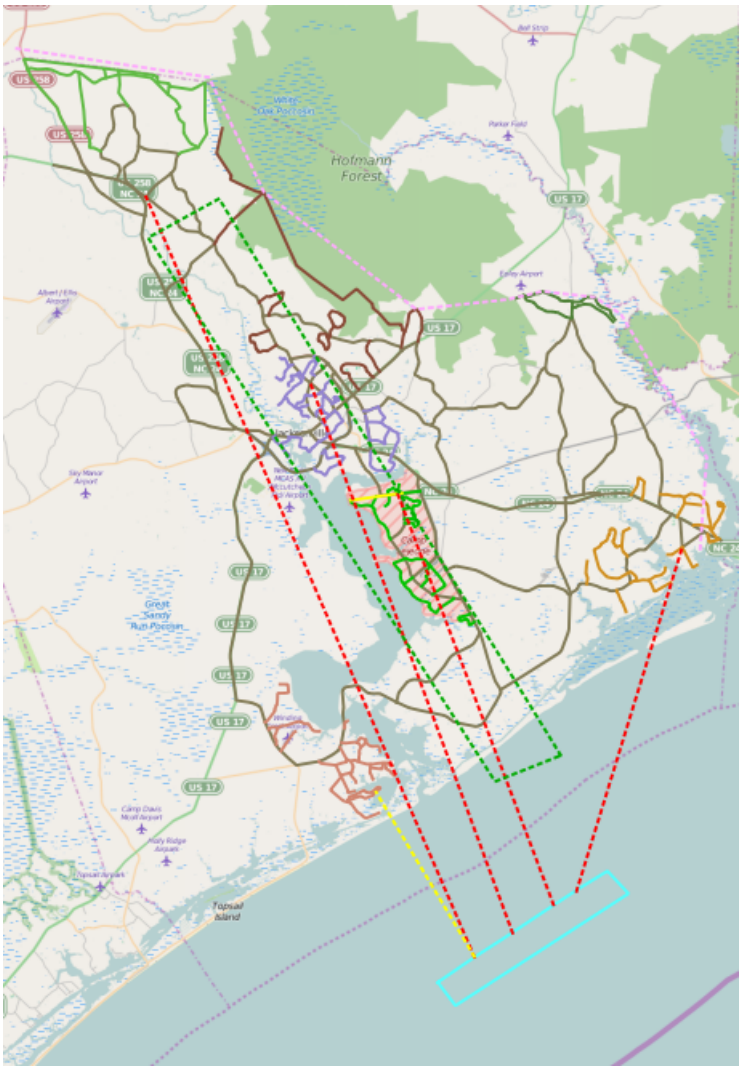


Figure 3.5: Platoon Deployment Areas and Humvee/Aircraft Routes for the Military Simulation

3.4 Validation Simulation

GAPR was originally coded in ONE, so the original source code was available. However, between the time that GAPR was first coded and this experiment ONE transitioned from version 1.4.1 to version 1.5.1. Further, Java transitioned from version six to version seven. The structure of the modules in ONE changed, and the MaxProp protocol included in ONE was updated. Since GAPR extends MaxProp and includes comparators that were built for Java six, several updates to the original GAPR code were required. Before conducting additional analysis, verification that the changes did not affect the behavior of GAPR was required.

Validation Settings			
Scenario time	43,200 s (12 hours)	Pedestrian Groups Nodes per Group	2 groups 40 nodes
Warmup time	1,000 seconds	Automobile Groups Nodes per Group	1 group 40 Nodes
Update Interval	0.1 s	Number of Trams	3 Trams 2 cars per tram
Tram Radio Range	1000 meters	Tram wait time	0 to 30 seconds
Pedestrian and car radio range	10 meters	Pedestrian and car wait time	0 to 120 seconds
Buffer Sizes	5 MB, 10 MB, 25 MB, 50 MB		
Transmit Speeds	.125 MBps, .5 MBps, 1 MBps, 5 MBps, 10 MBps		
Seeds	1, 2, 3, 4		

Table 3.2: Validation Settings

Vector was fundamentally based on the work done by Kang and Kim in [1]. However, Kang and Kim did not specify implementation details and conducted simulation in NS-2. The concepts from [1], namely, limiting replication of messages to nodes that are traveling in a parallel directions, are coded into a new implementation in Java for use with ONE. This implementation includes certain aspects of MaxProp and extends the Active Router super-class, so validation of this implementation of Vector is also required.

Both protocols are validated using the Helsinki scenario with settings matching those of the original GAPR scenario as denoted in Table 3.2. The results of the original GAPR work are available, and the results of this simulation should be identical. Validation of Vector is completed using the same scenario through a combination of logging and simulation output analysis.

Logs were produced by adding system.out calls to the source code and piping the console output to a log file. At each node encounter the result of Equation 3.1, the current node heading as calculated by the node, Equation 3.2, the angle of incidence between the two nodes, and Equation 3.3, the limit imposed on the number of messages to be exchanges, was logged, along with the number of messages actually transferred. Assertions were also included in the source code that throw errors that throw errors when logical inconsistencies occur. There were tens of thousands of entries in the log files, so a script was created to validate the logged data.

Helsinki Scenario Settings			
Scenario time	86,400 seconds (1 day)	Pedestrian Groups Nodes per Group	2 groups 40 nodes
Warmup time	10,800 seconds	Automobile Groups Nodes per Group	1 group 40 Nodes
Update Interval	0.1 s	Number of Trams	3 Trams 2 cars per tram
Radio Interface model	simple broadcast	Pedestrian Speed	0.5 - 1.5 m/s
Pedestrian and car radio range	10 meters	Pedestrian and car wait time	0 to 120 seconds
Tram interface range	1000 meters	Car Speed	2.7 - 13.9 m/s
Tram buffer Modifier	10x pedestrian size	Tram speed	7 - 10 m/s
Message Size	500k to 1MB	Tram wait time	10 to 30 seconds

Table 3.3: Control Settings Common to All Helsinki Simulations

3.5 Helsinki Simulation

More exhaustive simulation using the Helsinki Scenario is called for, as it is a popular model used widely throughout literature in this field. The data derived from additional simulation rounds can be leveraged to compare GAPR, GAPR2, and Vector to other work and future protocols simulated in ONE. As discussed above, the Helsinki Scenario is designed to model specific nodes, namely pedestrians, automobiles, and trams, as they travel through the city of Helsinki. Modifying the settings of these nodes would be to invalidate the realism which is the very reason for using the model. Therefore, throughout the expanded Helsinki simulations node mobility settings are maintained. Likewise, trams always have a buffer size 10 times larger than pedestrians and cars, representing a data mule system installed in the trams to facilitate the city-wide deployment of such a network. These settings and other control settings that are not varied between runs are outlined in Table 3.3.

Dependent variables open to investigation during extended Helsinki simulation include buffer size, transmission speed, and message generation rate. Two sets of simulations for each protocol are run in order to fully iterate over an interesting range of dependent variables. The first set of simulations examines changes in delivery ratio, overhead ratio, and latency across seven buffer sizes with a high and low message generation speed and a constrained and unconstrained data transmission rate.

The second set of simulations iterates over six transmission rates with constrained and unconstrained buffer size and the same high and low message generation speeds. The low message

Helsinki Simulation, Set 1						
Buffer Sizes	4M	6M	9M	14M	22M	35M
RNG Seeds	1	2	3	4	5	
Message Generator Speeds	10, 14	25, 35				
Transmission Speeds	250k	2M				

Table 3.4: Helsinki Simulation Set 1: Effects of Buffer Size

Helsinki Simulation, Set 2						
Transmit Speed	125k	250k	500k	1M	1.5M	2M
RNG Seeds	1	2	3	4	5	
Buffer Sizes	10M		50 M			
Message Generator Speeds	10-15 messages per second		25 - 35 Messages per second			

Table 3.5: Helsinki Simulation Set 2: Effects of Transmit Speed

generation speed is the default ONE value of one message per 25 to 35 seconds, and the high message generation speed is one message per 10 to 14 seconds. Constrained transmission speed is 250kBps, unconstrained transmission speed is 2MBps. Constrained buffer size is 15MB, unconstrained buffer size is 100MB. Note that transmission speeds are in bytes per second, not the standard measure of bits per second.

The values of the dependent variables iterated over during the first set of simulations described above can be found in Table 3.4, and the values for the second set of simulations can be found in Table 3.5. Constrained and unconstrained buffer sizes and transmission rates are derived from the observations made during the validation experiment. In each set, every combination is run five times using different random number generator seeds.

3.6 Random Mobility Simulation

The objective here is to show the performance of the protocols across a range of mobility and node density[6] settings when the ability to predict the movement of other nodes is constrained. Using the random walk mobility generator and a open scenario grid results in random node movement, negating the ability of nodes to successfully predict the future behavior of other nodes. To determine how node speed affects performance a range of speeds is required. Random

[6]Note that since the simulation grid size is constant, the terms node density, network size, and number of nodes refer to the same configuration setting, group.numberOfHosts, which determines the number of nodes in a group.

walk is also highly affected by the maximum duration setting, which determines how far a node will travel before changing direction. This equates to the rate which nodes travel away from their starting locations. Finally, these simulations provide a perfect environment to examine how altering the size of the network effects performance, since nodes will be spread out over the simulation grid vice constrained to certain paths.

To alter the maximum duration setting of the random walk mobility generator the source code of the RandomWalk class inside the mobility package of ONE is changed. Therefore, these simulations are done in 3 batches, each batch with a different maximum duration setting. The number of nodes and node speed is varied in each batch of simulations. The number of nodes ranges from 5 nodes to 50 nodes and the node speed ranges from 1 meter per second to 50 meters per second. Each combination is run for 5 random number generator seeds. These settings are depicted in Table 3.6.

Previous work and the validation simulations in this work provided a reasonable range of dependent variables to iterate over for the extended Helsinki simulations. There is no such data to inform the starting values for the random walk scenario, so several runs using MaxProp and Epidemic were conducted over large ranges of variables from 4 nodes to 200 nodes and from speeds of 1 meter per second to 100 meters per second. The results were used to determine where performance becomes asymptotic, and a range of values prior to the asymptotic phase are simulated. The ranges of the dependent variables can be found in the bottom three rows of Table 3.6.

Controls for this simulation include the size of the scenario grid which is maintained from Helsinki Scenario at 4,500 meters by 3,400 meters. A transmit speed of 2 MBps and a buffer size of 35 MB is also consistant across all simulations. These settings and other simulation configurations are also outlined in Table 3.6.

3.7 Military Simulation

The military simulation is meant to realistically portray the scenario described above. A critical element of that realism is the use of the actual topography surrounding Camp Lejeune. This is supported by the map-based mobility generators in ONE, but the only map format accepted is WKT. Each platoon is bounded by a different WKT file which is a subset of the Humvee layer depicted in Figure 3.4; thus each of the smaller areas depicted in Figure 3.5 is a separate WKT file. The aircraft also require a separate WKT file that defines their routes. All WKT

Random Mobility Simulation Settings			
Scenario time	86,400 seconds	Simulation Grid Size	4,500 m x 3,400 m
Warmup time	10,800 seconds	Buffer Size	35 MB
Update Interval	0.1 s	Wait Time	30 - 60 seconds
Transmission Speed	2 MBps	Transmission Range	100 meters
Interface Model	Simple Broadcast	Message TTL	100 minutes
Message Generation Speed	1 per 25-35 s	Message Size	200 kB to 1 MB
Maximum Duration Settings: 250 meters, 500 meters, and 1000 meters			
Node Speeds: 1 m/s, 2 m/s, 4 m/s, 7 m/s, 18 m/s, 25 m/s, 35 m/s, and 50 m/s			
Size of Network: 5 nodes, 10 nodes, 15nodes, 20nodes, 30nodes, 40nodes, and 50 nodes			

Table 3.6: Random Mobility Simulation Settings

files are required to be set to the same coordinate system and overlay the same base layer. This complicates creating WKT files that accurately model the topographical data, and necessitates the used of a GIS tool.

Quantum Geographic Information System (QGIS), [47], is a mapping tool for displaying and manipulating highly accurate GIS data. It also supports building vector-based layers for maps and ensures all layers are tied to the same grid reference system. To build the maps for this scenario the OpenLayers plug-in for QGIS was used to import a Open Street Maps (OSM) [48] base layer. Additional layers were built on top of the OSM layer, one layer for each group of nodes. Figure 3.5 is actually the final rendering with all layers displayed.

ONE only accepts WKT elements that are LINESTRINGs and POINTs. New layers in QGIS were built as vector shape-files, meaning each road is a series of map nodes (or vertexes). Each vector can be converted to WKT using the SimpleWKT plug-in, or the entire layer shape-file can be converted into WKT and exported using QGIS export functions. Since ONE requires all WKT files to form a fully connected graph, we exported each vector as it was created to a WKT file and ensured that the vectors shared common points. Note that when constructing the WKT files, each LINESTRING within a single file needs to be connected to a previous LINESTRING, and at least one point must be shared between two WKT files, such that when all files are combined a single connected graph is formed.

To ensure a connected graph, the Humvee layer layer was built first and includes all the major roads around Camp Lejeune. Each road is a series of LINESTRING segments that intersects at points shared between the segments. Each platoon layer was then built on top of the Humvee

Military Simulation Message Generator Settings			
Simulation Duration	86,400 seconds (24 hours)		
Warmup Time	10,800 seconds		
Simulation Interval	0.1 seconds		
Interface Model	Simple broadcast		
Group Name	Marines	Humvees	Ships
Group ID	M#-	H-	S-
Msg Gen Rate	5-10 sec	10 - 20 sec	25 - 35 sec
Message Size	250k - 500k	500k - 1M	500k - 1M

Table 3.7: Simulation Settings and Message Generator Settings for Military Simulation

layer, and incorporates whatever LINESTRINGS from the Humvee layer pass through the platoon area. Finally the aircraft layers were built, each including a point from a previous layer. The end state is a separate WKT file for each platoon, the Humvees, each pair of helicopters, the LCAC, the drones, and the ships.

The simulation is run using the map based mobility generator for the marine platoons and map route based mobility generator for the aircraft and the ARG. Each group from Table 3.1 is assigned its respective WKT file and attributes as a separate group of hosts in the ONE simulation configuration file. The simulation duration is 86,400 seconds, with a warm-up period of 10,800 seconds and a 0.1 second update interval. All nodes use a simple broadcast interface model. Three message generators are used, one for each group, as shown below in Table 3.7.[7] The group IDs are used to label the nodes during simulation runs when the GUI is used, a image of which is shown in Figure 3.6.

[7] Aircraft do not generate messages organically, but transfer messages generated by other nodes.

ONE Simulator running the Military Scenario.

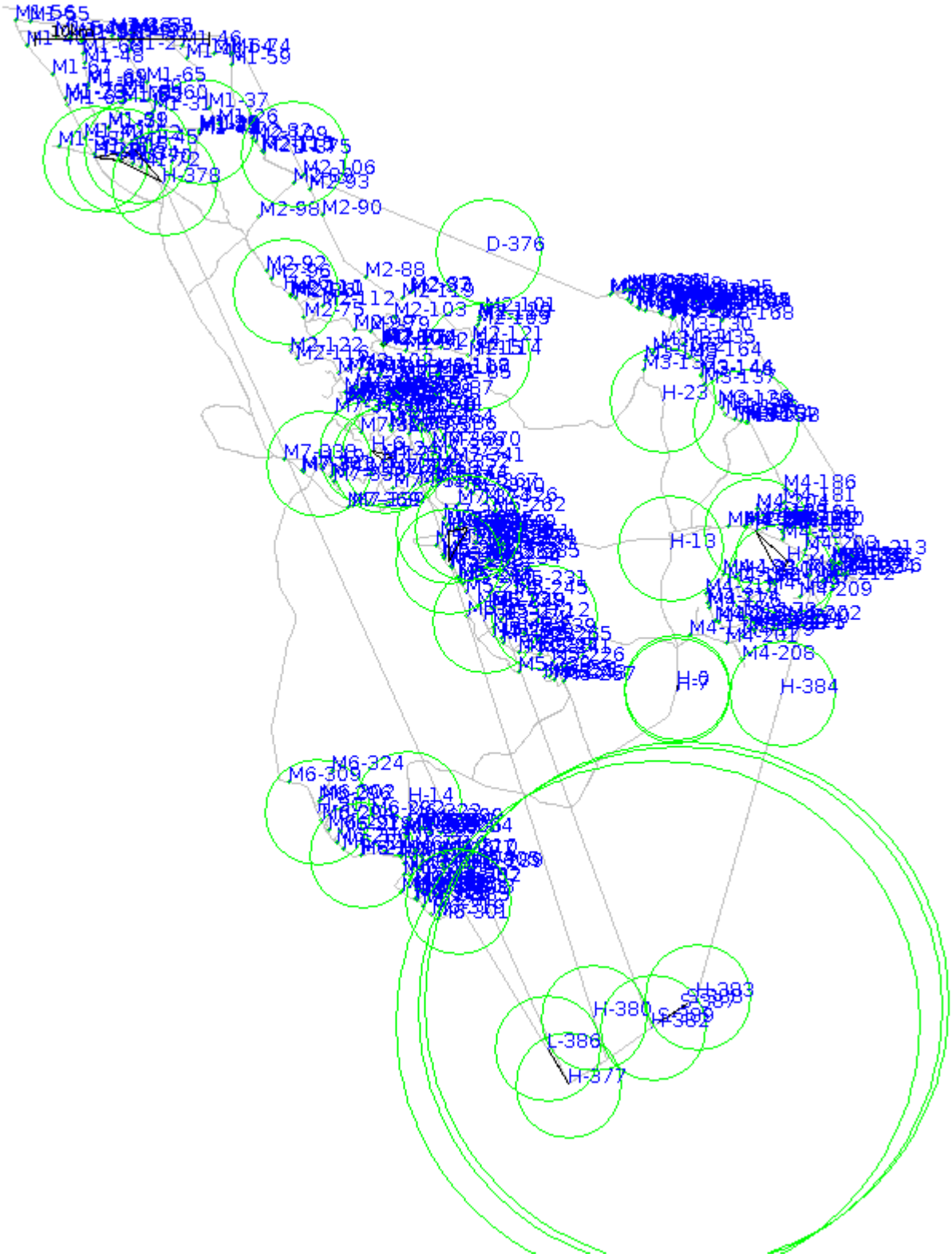


Figure 3.6: Military Simulation Running in ONE.

CHAPTER 4:
Analysis

The simulations in this work generate a lot of data. The purpose of this chapter is to clearly present as much data as possible, in as condensed a format as possible, and to provide context and analysis of the data presented. Analysis of protocol performance is limited in scope to the simulations of the scenario being analyzed. General conclusions and conclusions regarding multiple scenarios are reserved for Chapter 5. The primary protocols of concern in this work are GAPR, GAPR2, MaxProp, and Vector; these are occasionally referred to as the primary protocols.

Data visualization is accomplished primarily through the use of line graphs. Tables of the aggregated data are also included as appendices for those interested. The raw data is not included due to the size of the datasets, because graphs and analysis are all based on the aggregated data, and because we feel that further analysis, recreation, or validation of this work can be accomplished with the aggregate data provided. For data presentation, we find that line graphs are the simplest, clearest method of changes in the measures of performance across a range of dependent variables.

This chapter is broken into four primary sections, one for the validation scenario, one for the Helsinki scenarios, one for the random mobility scenarios, and one for the military simulation. Within each section is a brief review of the simulation, followed by a data presentation section. Analysis of the measures of performance follows data presentation.

4.1 Validation Simulations

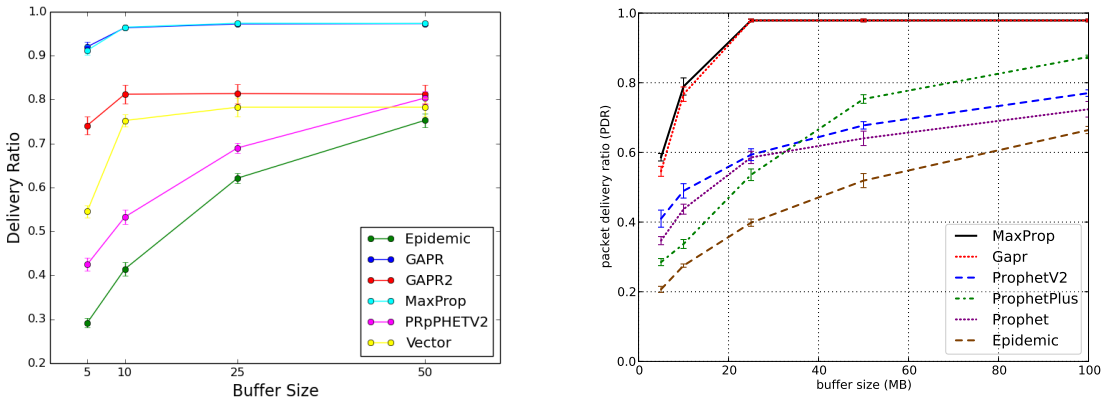
These simulations are conducted using the same configuration settings as the original GAPR simulations conducted by Rohrer et. al. in "Geolocation Assisted Predictive Routing (GAPR) Protocol for Heterogeneous DTN Mobility Patterns," [2], in order to recreate and validate the original work. These validation simulations provide the foundation for future simulations of GAPR throughout this work and show that the updates to the GAPR source code did not alter the behavior or performance of the protocol.

In validating GAPR we are fortunate to have access to the original figures. Therefore we graph the results of these simulations using the same parameters as the graphs in the original work

and display the original graphs from [2] alongside the graphs produced from these simulations. Vector is included in these simulations even though there is no previous data in [1] to directly compare to our implementation. To validate vector we used code logs collected while the simulations were executed as described in Chapter 3. We also used code execution-time logging to validate GAPR, but the figures described below provide clear visual validation.

4.1.1 Data Presentation

Here we present the figures that appeared in the GAPR paper, Figure 4.1b and Figure 4.2b, alongside the results of our validation scenarios, Figure 4.1a and Figure 4.2a. The first set of simulations, 4.2a and 4.2b, show delivery ratio over a set of buffer sizes from 5 MB to 50 MB. The second set, 4.2a and 4.2b, show delivery ratio over a range of transmission speeds. Note that transmission speeds are in bytes per second, not the typical bits per second. For simple conversion, .250 mBps is equivalent to 2 Mbps.



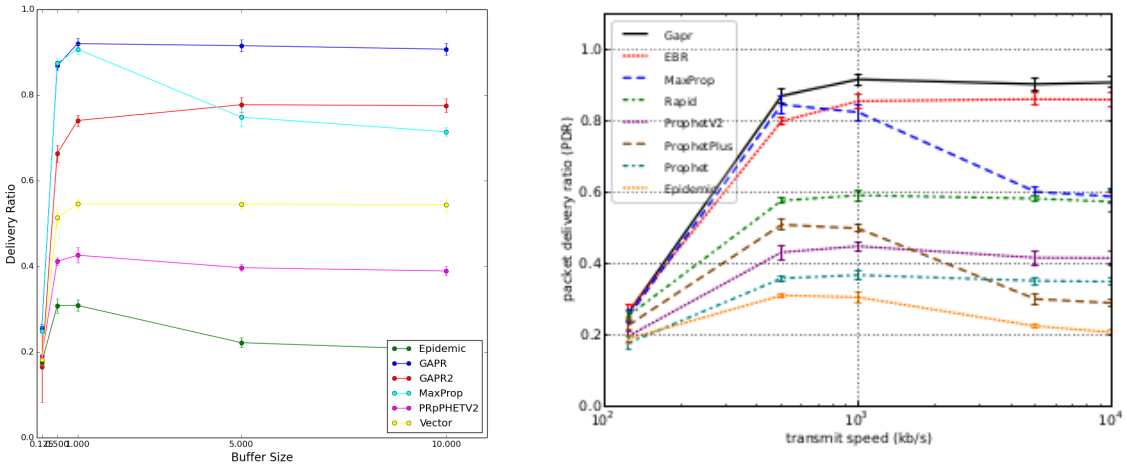
(a) Validation Simulation

(b) Original findings from ref [2]

Figure 4.1: Validation Simualtion Vs. Original Work, Delivery Ratio vs Buffer Size

4.1.2 Analysis of Validation Simulations

The performance of GAPR in the validation simulations is identical to the performance of GAPR in the original work. At the 5 MB mark in the delivery ratio of GAPR is 92% while the delivery ratio of MaxProp is 91% as shown in Figure 4.1a. GAPR in the original work, as shown in Figure 4.1b, displays the same delivery ratio of 91%. MaxProp delivery ratio with a 5 MB buffer size appears slightly lower in the original graphs then indicated by this experiment, which could be due to the updates to the MaxProp code during ONE and Java updates, but is



(a) Validation Simulatio

(b) Original findings from ref [2]

Figure 4.2: Validation Simualtion Vs. Original Work, Delivery Ratio vs Buffer Size

beyond the scope of this thesis. In both graphs delivery ratio for GAPR and MaxProp reach a limit of approximately 97% delivery ratio and become asymptotic.

In the second set of graphs, Figures 4.2a and 4.2b, the delivery ratio of GAPR remains steady above 90% while MaxProp begins to decline when transmission speeds increase past past 1 MBps. The X axis is plotted in a logarithmic scale in both graphs. Also EBR is not plotted in 4.2a; the other protocol with delivery ratio above 80% in that plot is GAPR2.

4.2 Helsinki Simulation

The Helsinki Simulations are run to realistically model the behavior of DTN protocols in a urban environment. Two sets of simulations are conducted, one with increasingly constrained buffer sizes, and one with increasingly constrained transmission speeds.[8] Simulations are also conducted under high and low network traffic loads, where high network load is modeled by setting the message generator to generate a new message every 10 to 14 seconds, and low network load is modeled by generating one message every 25-35 seconds. Message generation, message size, and buffer sizes are scaled to appropriate levels for simulation. The size of each message is between 500 kB and 1 MB as determined by a uniformly-distributed RNG. Thus with an average size of .75 MB per message, a node can hold, on average, 4 messages per three MB of buffer space.

[8]Note that transmission speeds are again measured in MBps vice the more common Mbps.

4.2.1 Helsinki Simulation Set 1: Constrained Buffer Size

The first set of Helsinki Simulations are run over a range of buffer sizes from 4 MB through 35 MB. Two transmission speeds are simulated, 250 kBps (2 Mbps) and 2 MBps (8 Mbps). Under the constrained setting of 250kBps, a node with the largest buffer size of 35 MB can carry an average of 46 messages, but would require an encounter duration of 140 seconds to transmit all messages. With unconstrained transmission speed, the same node can transmit all stored messages in 17.5 seconds. The high network traffic load setting generates 2.5 times more network traffic then the low traffic setting.

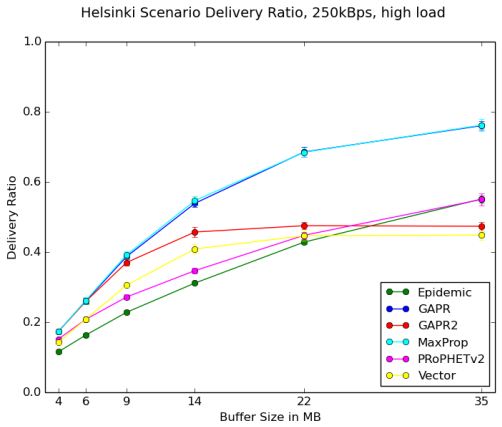
One item to consider when simulating nodes constrained by their buffer size is that nodes always make room for messages generated locally. This can be significant at very constrained buffer settings and high network load, as two 1 MB messages generated locally cuts in half the space available to carry messages forwarded by other nodes. Finally, in some simulations the behavior of Epidemic and PRoPHET has not become asymptotic at the 35 MB high buffer setting, but the performance of the primary protocols does show asymptotic behavior in nearly every simulation by the 35 MB buffer mark.

Data Presentation

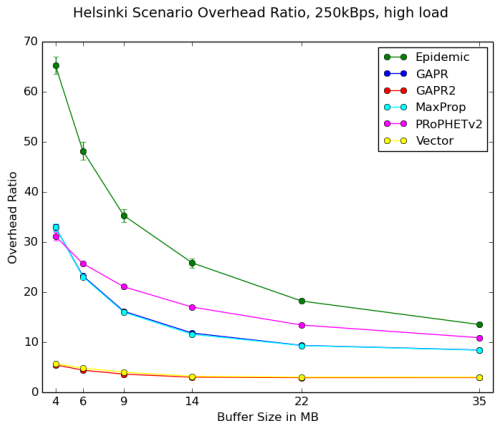
The entire set of Helsinki simulations are presented as graphs, each graph representing 180 simulation runs. The graphs over constrained buffer sizes are included below as Figures 4.3a through 4.6c. Within that set of figures, the first column shows the effects of buffer size on delivery ratio, the second column shows effects on overhead, and the third column shows effects on latency. The first six graphs, or top two rows, show performance under high network load; the bottom rows show the same measurements under low network load. Rows one and three show performance with transmission speed constrained to 250 kBps, and rows two and four show the results of allowing 2 MBps transmission speeds.

Analysis of Delivery Ratio in Helsinki Simulation set 1

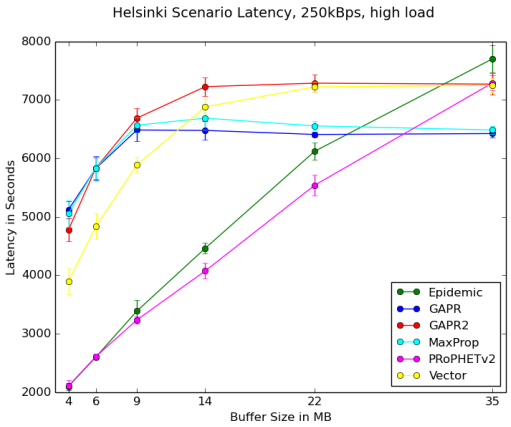
Figures 4.3a, 4.4a, 4.5a, and 4.6a show the effect of increasing buffer sizes on delivery ratio. At nearly every measurement point across all simulations the delivery ratio of MaxProp and GAPR are within one another's 95% confidence intervals. The only case where this is not true is when network load is low and transmission speed is high, as shown Figure 4.6c. Here, GAPR has a 9.2% higher delivery ratio then MaxProp when buffer size is 4 MB and a 2.3% higher delivery ratio at the 6MB buffer mark. Also seen in 4.4a, GAPR displays higher delivery ratio between the 6 MB and 14 MB buffer sizes.



(a) Buffer Size Vs. Delivery Ratio



(b) Buffer Size Vs. Scenario Overhead

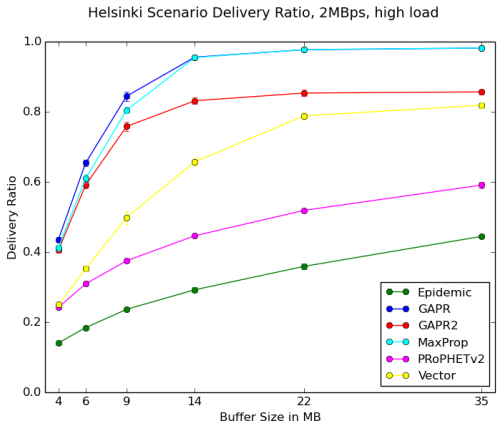


(c) Buffer Size Vs. Latency

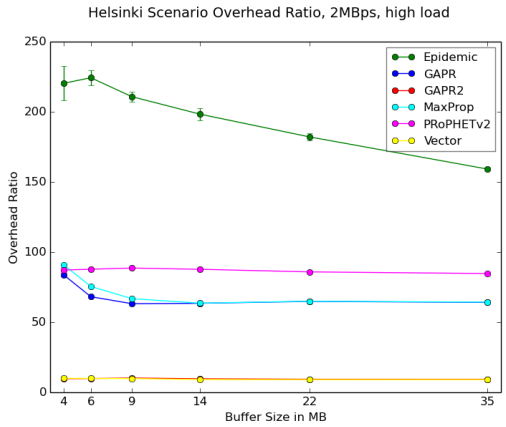
Figure 4.3: Helsinki Simulation Set 1, High Network Load and Small Buffers

The delivery ratio of GAPR2 and Vector also tend to be pegged to one another but with a larger difference between the two; in most simulations GAPR2 has 5% to 10% higher delivery ratio then Vector. However under the most constrained buffer settings and with unconstrained transmission speed as shown in Figure 4.4a, GAPR2 vastly outperforms Vector with respect to delivery ratio. This is also evident to a lesser degree with less constrained buffers (as a byproduct of reduced network traffic load) in Figure 4.6a.

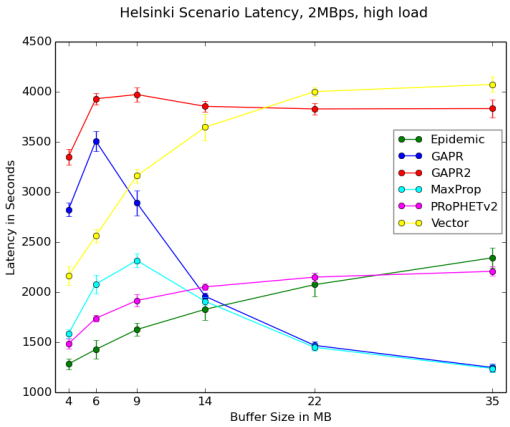
Both GAPR and MaxProp have higher delivery ratios then all other protocols across all Helsinki simulations. GAPR2 has a higher delivery ratio then Vector in all of these simulations. Keeping all other attributes constant, removing the transmission speed constraint vastly increases the delivery ratio of all protocols. When constrained by buffer size, increasing only the transmission speed results in a delivery ratio increase of, on average, 25% at the most constrained buffer



(a) Buffer Size Vs. Delivery Ratio



(b) Buffer Size Vs. Scenario Overhead

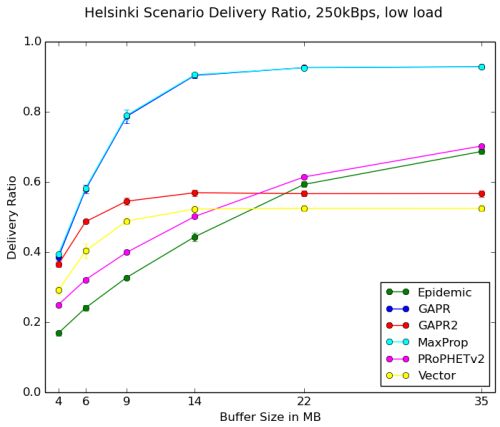


(c) Buffer Size Vs. Latency

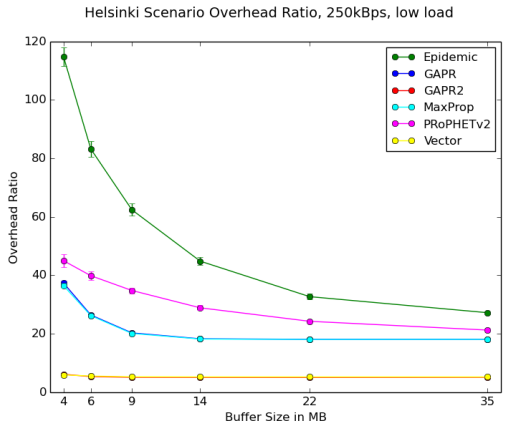
Figure 4.4: Helsinki Simulation Set 1, High Network Load and Large Buffers

setting and a 20% higher asymptotic value for GAPR and MaxProp. Under low network load this effect is also observed until GAPR and MaxProp reach above 95% delivery ratios and exhibit asymptotic behavior. The effect of increased transmission speeds on GAPR2 and Vector raises their asymptotic values by 38% for GAPR2 and 36% for vector. Further, the delivery ratio of GAPR2 and Vector appears to have a limit around 90% with 2 MBps transmission speed, where GAPR and MaxProp reach nearly 99%. The change in delivery ratio from 4.5a to 4.6a or between 4.3a and 4.4a show the effects of transmission speed as buffer size constraints ease.

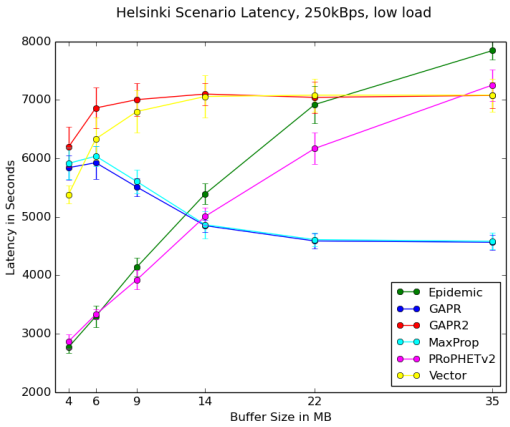
The clear trend here is that delivery ratio increases with larger buffer sizes, albeit with diminishing returns, until protocols reach a maximum delivery ratio. At this point performance gains become minimal, even as buffer size continues to increase. GAPR and MaxProp enter a asymp-



(a) Buffer Size Vs. Delivery Ratio



(b) Buffer Size Vs. Overhead



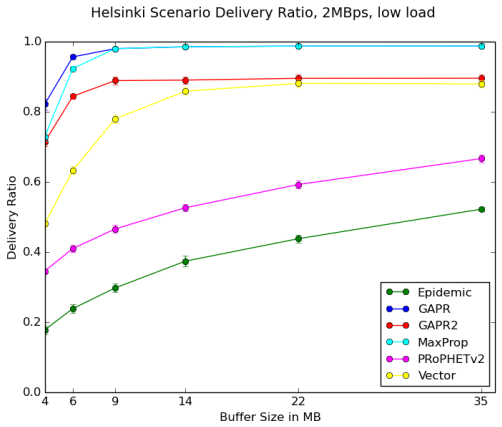
(c) Buffer Size Vs. Latency

Figure 4.5: Helsinki Simulation Set 1, Low Network Load and Small Buffers

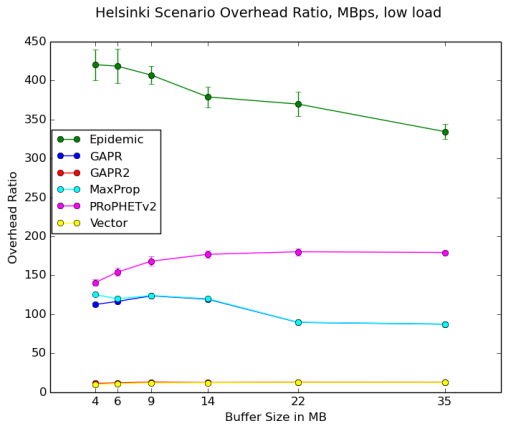
totic phase bounded by delivery ratios above 97% faster then other protocols and while biffer sizes are still constrained, indicating that their buffer management techniques are most effective. Further, the difference in the delivery ratio of Gapr/MaxProp and GAPR2/Vector shown by a comparison of Figures 4.4a and 4.5a indicates that MaxProp and GAPR are not as affected by the reduction in transmission speed. Both of these protocols use a threshold calculation that takes into account the average number of bytes transferred at each encounter, which combined with less restrictive replication appears to allow them to more successfully adapt to constrained transmission speeds then other protocols.

Analysis of Overhead in Helsinki Simulation set 1

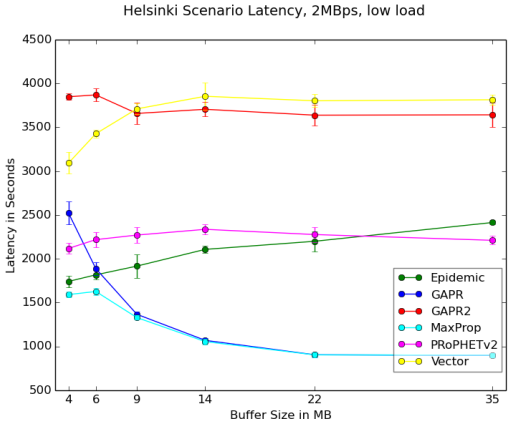
Figures 4.3b, 4.4b, 4.5b, and 4.6b show the effect of increasing buffer sizes on the overhead ratios of the protocols. The first and most striking observation here is the large gap in overhead



(a) Buffer Size Vs. Delivery Ratio



(b) Buffer Size Vs. Overhead



(c) Buffer Size Vs. Latency

Figure 4.6: Helsinki Simulation Set 1, Low Network Load and Large Buffers

incurred by Vector and GAPR2 as compared overhead incurred by MaxProp and GAPR. When buffer size is highly constrained, the overhead ratio of GAPR2 and Vector is a mere tenth that of GAPR and MaxProp. When buffers are larger and the behavior of the protocols levels out, the overhead ratio of GAPR and MaxProp remains six to eight times larger then GAPR2 and Vector. In the two scenarios with 2 MBps transmission speeds, 4.4b and 4.6b, where protocols reach a reasonable delivery ratio, the overhead of GAPR2 and Vector is 9.15 and 12.8, where the overhead ratio of GAPR and MaxProp is 64 and 87.4, respectively.

GAPR2 and Vector reduce overhead by limiting the number of messages forwarded to nodes traveling in the same direction. This limits needless replication of messages between nodes that are likely to encounter the same new nodes in the near future. Because of the nature of the urban movement model implemented in the Helsinki Scenarios, many nodes encounter one another

while transiting corridors that limit the directional headings of those nodes. The replication limiting mechanism of GAPR2 and Vector is amplified under such conductions. Interestingly, this overhead reduction is also shown to be flat, even under very constrained buffer sizes, where other protocols overhead ratio grows exponentially.[9]

Analysis of Latency in Helsinki Simulation set 1
Figures 4.3c, 4.4c, 4.5c, and 4.6c show the effect of increasing buffer sizes on latency. GAPR2 and Vector are the poorest-performing protocols in terms of latency, and as buffer size increases the latency of GAPR2, Vector, Epidemic, and PRoPHET increases, while GAPR and MaxProp latency falls (with the exception of the most constrained scenario, high network load and slow transmission speed, 4.3c, where the latency of all protocols increases). After the latency curves flatten, GAPR2 and Vector show 40% higher latency then MaxProp and GAPR when transmission speed is slow and fourfold higher latency when transmission rate is high.

Under high network load and slow transmission speeds, GAPR2 and Vector average 7,260 second latency, while MaxProp and GAPR average 6,450 seconds, a reduction greater then 10%. With fast transmission speeds the latency incurred by GAPR2 falls to 3,800 seconds, while the latency of GAPR and MaxProp falls to 1,240 seconds. The most significant difference in latency is noted in the least constrained case, where network load is low and transmission speed is high, shown in Figure 4.6c. Here the latency of GAPR and MaxProp is just below 900 seconds, while the latency of GAPR2 is four times higher at over 3,600 seconds.

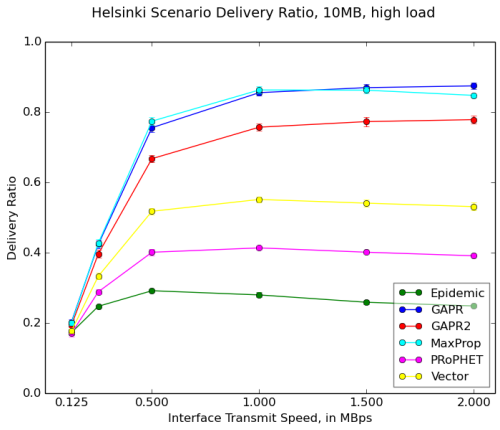
4.2.2 Helsinki Simulation Set 2: Constrained Transmission Rate

The second set of Helsinki Simulations are run over a range of transmission speeds from 125 kBps (.125 MBps as labeled on the graphs, or 1 Mbps in the common unit of measure for in networking) through 2 MBps. Similar to the simulations over a range of buffer sizes that were run over constrained and unconstrained transmission speeds in the previous set, this set of simulations is run over two buffer sizes, 10 MB and 50 MB. The 10 MB buffer size is small enough that it does not allow pure flooding, but not so small as to mask the effects of varying transmission speeds. The 50 MB buffer size represents a unconstrained buffer size. 50 MB is large enough to carry 66 messages and requires a 400 second encounter to fully transmit all messages with a transmission speed of .125 MBps, or a 25 second encounter at 2 MBps. These simulations are also run over the same two network load settings previously described.

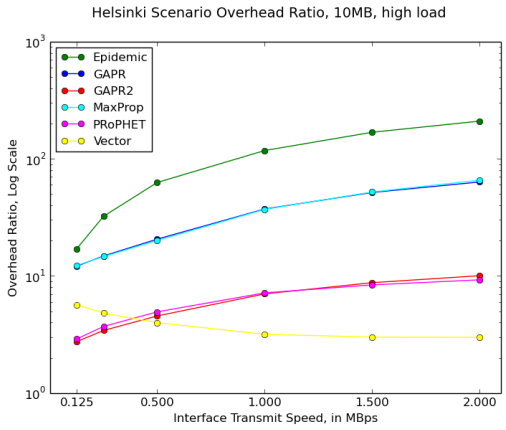
[9]Note that the graphs in this section show buffer size increasing; observing the curves in reverse highlights the effect of shrinking buffers. The exponential decay in overhead ratio as buffer size increases thus represents exponential growth as buffer size decreases.

Data Presentation

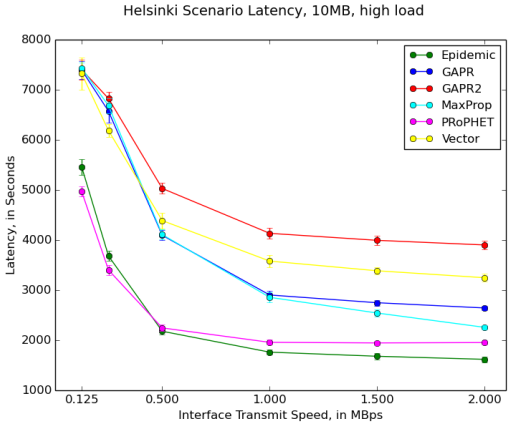
The graphs over constrained transmission speeds are included below as Figures 4.7a through 4.10c. The layout of these graphs is the same as the previous set; the first column shows the effects of transmission speed on delivery ratio, the second column shows effects on overhead, and the third column shows effects on latency. The first six graphs showing performance under high network load and the bottom six showing high network load and the last six low network load, rows one and three showing the 10 MB buffers and rows two and four showing 50 MB buffers.



(a) Transmission Speed Vs. Delivery Ratio



(b) Transmission Speed Vs. Scenario Overhead

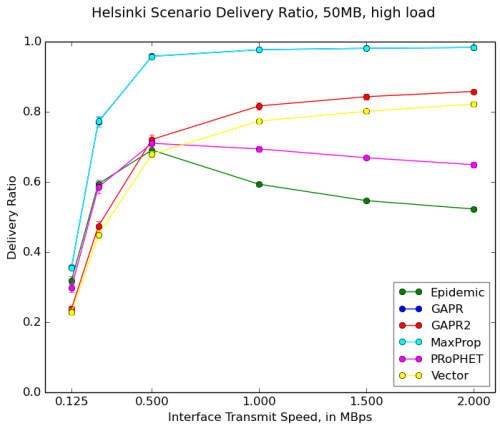


(c) Transmission Speed Vs. Latency

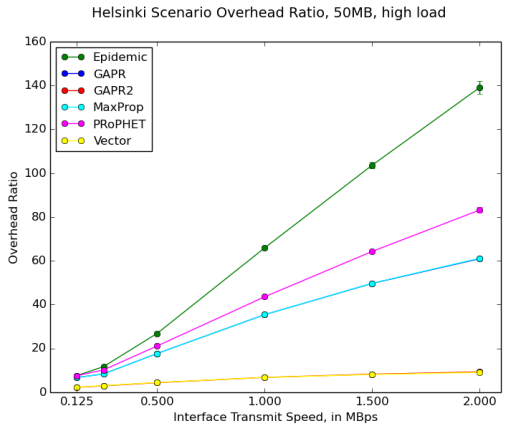
Figure 4.7: Helsinki Simulation Set 2, High Network Load and Small Buffers

Analysis of Delivery Ratio in Helsinki Simulation set 2

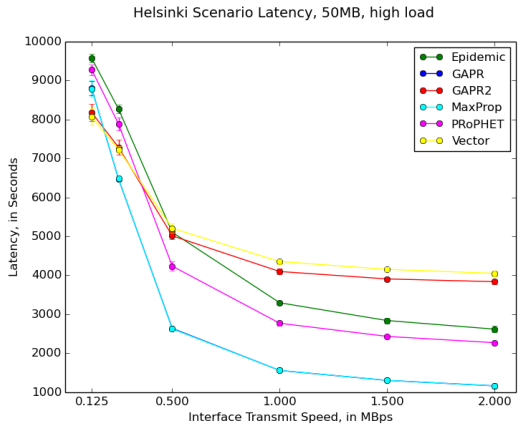
Figures 4.7a, 4.8a, 4.9a, and 4.10a show the effect of increasing transmission speeds on delivery ratio. As expected, a similar proportional trend that meets with diminishing returns is observed



(a) Transmission Speed Vs. Delivery Ratio



(b) Transmission Speed Vs. Scenario Overhead

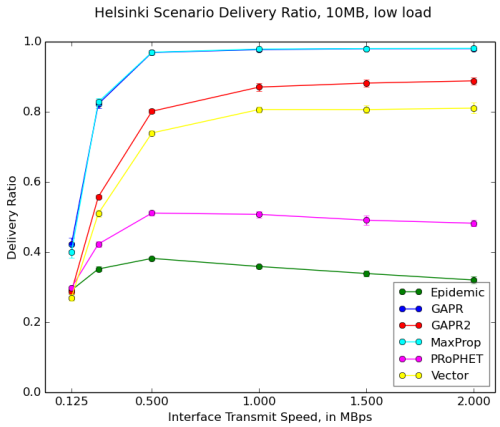


(c) Transmission Speed Vs. Latency

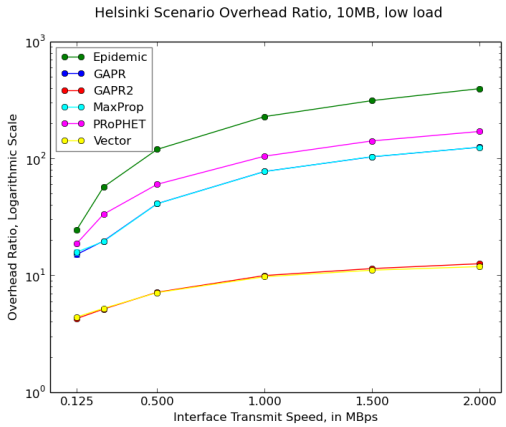
Figure 4.8: Helsinki Simulation Set 2, High Network Load and Large Buffers

between increasing transmission speeds and higher delivery ratios. This holds for the primary protocols, but interestingly and somewhat counterintuitive, higher transmission rates cause the delivery ratio of Epidemic and PRoPHET to fall, likely because they are constrained by overhead. When buffer size is unconstrained, the delivery ratio of MaxProp and GAPR quickly reaches above 97% as shown in Figure 4.10a, but encounters diminishing returns, and slowly approaches what appears to be a upper limit between 98% and 99%. The upper limit for GAPR2 and Vector in this scenario approaches 90%.

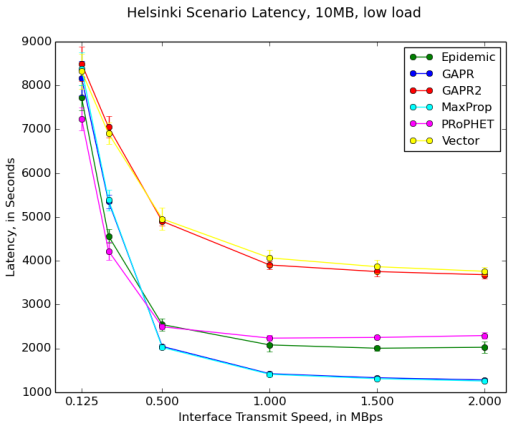
When buffer size is limited and transmission speed is high, as represented best in Figure 4.7a, the delivery ratio of MaxProp actually begins to decline, while GAPR remains asymptotic. The delivery ratio of MaxProp falls by .058% between 1 MBps and 1.5 MBps, where the 95% confidence interval is .0099, marking this as the beginning of statistically significant decline.



(a) Transmission Speed Vs. Delivery Ratio



(b) Transmission Speed Vs. Overhead

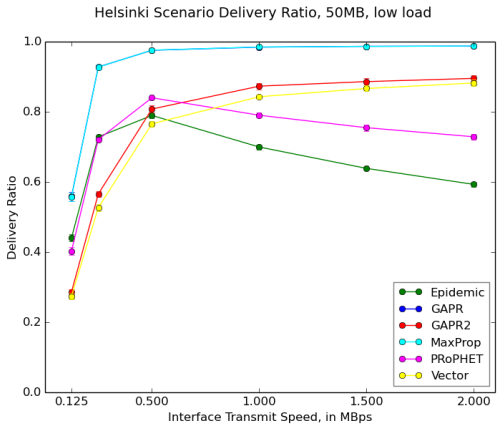


(c) Transmission Speed Vs. Latency

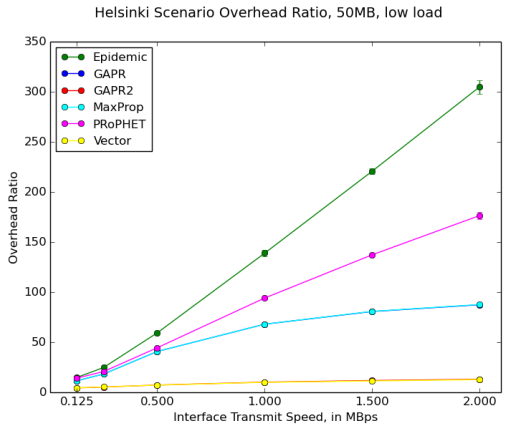
Figure 4.9: Helsinki Simulation Set 2, Low Network Load and Small Buffers

This continues between 1.5 MBps and 2 MBps transmission speed intervals, where MaxProp delivery ratio falls by 1.5%. Over the same intervals GAPR delivery ratio continues to rise by 1.4% and 0.50%, respectively. In the same scenario GAPR2 delivery ratio is nearly 35% higher then Vector after settling, the most significant delivery ratio difference between GAPR2 and Vector noted across all Helsinki simulations.

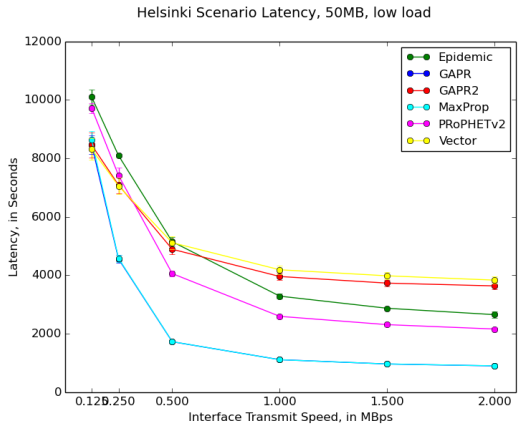
Also interesting is the difference in the behavior of the primary protocols as compared to that of Epidemic and PRoPHETv2 between the high network load, large buffer scenario shown in Figure 4.8a, and the low network load, small buffer scenario shown in Figure 4.5a. Network load decreases by 2.5 times and buffer size is decreases five-fold between the scenarios depicted in these figures. The delivery ratio of PRoPHETv2 and Epidemic fall by over 20%, while the



(a) Transmission Speed Vs. Delivery Ratio



(b) Transmission Speed Vs. Overhead



(c) Transmission Speed Vs. Latency

Figure 4.10: Helsinki Simulation Set 2, Low Network Load and Large Buffers

delivery ratio of MaxProp and GAPR is unaffected,[10] and the delivery ratio of GAPR2 and Vector actually increases by 8% and 6%, respectively.

Analysis of Overhead in Helsinki Simulation set 2

Figures 4.7b, 4.8b, 4.9b, and 4.10b show the effect of increasing transmission speed on overhead ratio. Note that Figures 4.7b and 4.8b are graphed using a logarithmic Y axis scale. In every case, increasing transmission speed results in higher overhead ratios. Across the two buffer sizes simulated, overhead for MaxProp and GAPR is lower when larger buffers are utilized, while the overhead of Vector and GAPR2 remains largely unaffected.

GAPR2 and Vector incur significantly less overhead in every simulation, and their overhead

[10] MaxProp and GAPR are already behaving asymptotically at this point.

ratio also grows are a much slower rate. The maximum GAPR2 and Vector overhead is 12.9, where the maximum overhead of GAPR and MaxProp is 124.5. The rate of overhead growth for Vector and GAPR2 as transmission speed increases averages 3.9 for every MBps in transmission speed gained, where for MaxProp and GAPR the rate of overhead growth averages 36.36 per Mbps.[11]

Analysis of Latency in Helsinki Simulation set 2

Figures 4.7c, 4.8c, 4.9c, and 4.10c show the effect of increasing transmissions speed on latency. All protocols demonstrate exponential decay of latency as transmission speed increases. Changing network traffic load and buffer size has the least effect on the Latency of PRoPHETv2 and Epidemic, which have the lowest latency in the most constrained scenario as shown in Figure 4.7c. GAPR2 and Vector have the highest latency in every scenario, while MaxProp and GAPR that the lowest in all scenarios save the most constrained.

GAPR2 and Vector display the highest latency of all protocols in every simulation, with GAPR2 incurring particularly high latency in the most constrained scenario shown in Figure 4.7c. Here GAPR2 latency is 16.5% higher then Vector, 38.2% higher then GAPR, and 42.2% higher then MaxProp. In the least constrained scenario, where network traffic load is low and buffers are large, Vector and GAPR2 have similar latency at 3,834 seconds and 3,636 seconds, while the latency of MaxProp and GAPR is 901.0 seconds and is 989.5 seconds respectively, as depicted in Figure 4.10c.

4.3 Random Mobility Simulations

Random mobility simulations are conducted to examine protocol behavior when there are no patterns to node movement. This tests the adaptability of knowledge-based protocols to scenarios where contact schedules are difficult or impossible to predict (see Figure 2.1), making knowledge of past encounters is less helpful. These simulations are also conducted using a open plane, where nodes are able to move anywhere within the maximum bounds of the grid, as compared to map-based scenarios where nodes are generally restricted to a very small portion of the simulation grid defined by paths or routes. Because nodes are not forced to move along predetermined paths or constrained to a small subset of the simulation grid the effect of node mobility on protocol performance is more readily examined.

Two elements of node mobility are varied in this set of simulations, node speed and maximum

[11] Note that growth is more linear then it appears in graphs with a logarithmic Y axis scale.

duration. Maximum duration affects how likely nodes are to travel far from their origin; on average over a long, large simulation, nodes travel half the distance of the maximum duration then turn 180 degrees.[12] Thus node speed and maximum duration both play a part in the diffusion of nodes across the simulation grid, which is required to support high delivery ratios. The size of the network is also varied, from sparse networks that consist of only 4 nodes to relatively dense networks of 50 nodes.[13] Buffer size and transmission speed is maintained across all simulations at 35 MB and 2 MBps, respectively. The transmission range of nodes is also maintained at 100m, allowing each node to cover $31,415\text{m}^2$, or 0.205% of the simulation grid.

4.3.1 Data Presentation

Each set of random mobility simulations is conducted over a range of seven different node densities and nine different node speeds, for 63 combinations per duration setting, or 189 combinations of variables for each protocol. This makes graphing all combinations of dependent variables simulated as was done with the Helsinki simulations untenable. Therefore relevant graphs are referred to in each analysis section, and aggregated data is available as appendices 3 through 5 for the 250 m duration, 500 m duration, and 1000 m duration simulations, respectively.

Delivery Ratio under Random Mobility

Figures 4.11a, 4.11b, and 4.11c show delivery ratio across different durations settings for networks of 10 nodes, 20 nodes, and 50 nodes, respectively. Figures 4.12a, 4.12b, and 4.12c show more directly the effect on delivery ratio when duration is increases across a range of speeds, for several network sizes. Figure 4.13 shows the impacts on delivery ratio as the number of nodes increases.

In the first three figures asymptotic growth in delivery ratio is observed as node speed increases for all protocols. Increasing node speed or the random walk maximum duration setting yields increases in delivery ratio for all protocols simulated across all simulations. Increasing node speed from very low mobility, namely between 1 m/s to 14 m/s, tends to equate to rapid gains in delivery ratio, but as node speed continues to increase gains in delivery ratio become asymptotic. Continuing to increase node speed past 20 m/s does not result in significant gains to the

[12] A uniformly distributed random number generator is used to select within a range from 0 to 360 degrees for the change in direction and zero to the maximum duration as determined by the setting of the respective simulation set.

[13] Note that DTNs are generally sparse networks by definition; even the 'dense' networks in these simulation only cover 11% of the simulation grid at a time.

probability of delivery in these simulations regardless of network size or duration.

Increasing the maximum duration setting, which equates to nodes traveling further before pausing and choosing a new direction to travel in, raises both the starting delivery ratio and asymptotic maximum bound for every protocol. Figures 4.11a through 4.12c indicate that the effect of doubling the maximum duration is actually greater then the effect of doubling node speed. This is also well-illustrated between Figures 4.12a and 4.12b.

At low speeds it can be assumed that encounters are long for every message in the buffer can be exchanged. Further, GAPR and GAPR2 use the same buffer ordering scheme, and neither protocol is constrained by overhead. Therefore a reduction in the replication of messages negatively impacts delivery ratio in networks with slow-moving nodes more then when node speed is fast. This is a reasonable conclusion, as nodes with low mobility are less likely to encounter new nodes, making every encounter increasingly valuable to the dissemination of messages.

Overhead Ratio under Random Mobility

Figures 4.14a, 4.14b, and 4.14c show overhead ratio across different duration settings for networks of 10 nodes, 20 nodes, and 50 nodes, respectively. Similar patterns to those that developed in the Helsinki trials are recognizable. The overhead incurred by Vector and GAPR2 remains largely flat as nodes gain or loose speed, while GAPR and MaxProp have very high overhead when node speed (and delivery ratio) is low, but incur less overhead as speed and delivery ratio increase. Overhead reduction in in GAPR and MaxProp continues until reaching a lower bound where the overhead becomes constant, typically pegged between 4 and 8 times higher then Vector and GAPR2. Epidemic has incredibly high overhead, which dramatically increases as maximum duration is increased; the overhead of Epidemic runs off the graph scale in Figure 4.14c to reach more then triple the overhead of MaxProp and GAPR.[14]

Latency under Random Mobility

Figures 4.15a, 4.15b and 4.15c show the latency of protocols across the duration settings for the 10, 20 and 50 node networks. The latency across all protocols at 250m and 500 m maximum duration is highly variable. However, it is remarkable that the latency at these settings is relatively constant, not falling as node speed increases.[15]

[14]The graph scale of Figure 4.14c was allowed to better illustrate the difference in overhead of the main protocols. Including Epidemic would necessitate a logarithmic Y axis scale or force all main protocol overhead lines into a small portion of the graph.

[15]Note that latency in the 250 m duration, 10 node simulations was erratic, causing large 95 % confidence intervals from which no concrete data should be derived. The lines are left on the plot only illustrate the decreasing

The 1000 m results clearly show a exponential decay of latency as node speed increases. GAPR2 and MaxProp outperform the other protocols when duration is set to 1000 m, incurring a third of the latency of GAPR2 and Vector. At node speeds of 25 m/s, GAPR incurs a latency of 766 seconds, slightly approximately a third of the 2,206 second GAPR2 latency. At 50 m/s, the 662 seconds latency of GAPR remains roughly a third of the 1810 second latency incurred by GAPR2.

A substantial reduction in latency is also realized as network size increases. At the 1000 m duration setting, the latency of GAPR in a network of 5 nodes is 2,479 seconds, falling to 2,270 seconds with 10 nodes, 1,368 seconds with 20 nodes, and continues to fall until GAPR incurs only a 662 second latency when operating on a 50 node network. GAPR2 shows less reduction in latency as network size grows, starting at 2,368 seconds in a 5 node network and remaining fairly constant, with latency above 2,200 sec.

4.4 Military Simulations

The military exercise simulations, like the Helsinki simulations, are meant to realistically model a scenario that includes real-world equipment. The node attributes in these simulations are derived from the attributes of existing equipment, and the geography is derives from actual GIS data. Altering these attributes would invalidate the premise of the scenario. Therefore the given scenario is modeled in the simulator configurations, and the only element changed between runs is the random number generator seed used to control the mobility generators.

Data Presentation

Each protocol is simulated over a range of 10 random number generator seeds. The performance measurements for each protocol are averaged for all seeds, and a bar graph is presented for each performance metric. Each analysis section refers to the bar graph of interest. To aid in analysis several additional simulations were used to isolate specific elements of the larger simulation and are discussed in the analysis sections. Performance measurements of each protocol and a 95 % confidence interval for each measurement are also included in Table 4.1.

Data is also plotted visually as a series bar charts, Figure 4.16, to aid visual comparison for each of analysis section.

---

nature of latency as duration increases.

	Delivery ratio	Overhead	Latency
		Epidemic	
Mean	0.30677	147.67215	2.62424
CI(95)	0.0025077316	1.088912446	0.0409499353
		GAPR	height=6cm,
Mean	0.46241	26.62745	1.48717
CI(95)	0.0035486077	0.4148968728	0.0161999184
		GAPR2	
Mean	0.45674	7.11631	1.40549
CI(95)	0.0032884293	0.1580448598	0.0102968028
		MaxProp	
Mean	0.46373	26.59271	1.54074
CI(95)	0.0031829676	0.4441034123	0.0136242973
		PRoPHETv2	height=8cm,
Mean	0.33666	30.61692	1.74109
CI(95)	0.0018848897	0.8027403683	0.0107352513
		Vector	
Mean	0.37047	6.7053	1.51687
CI(95)	0.0052258659	0.1379096552	0.0112999751

Table 4.1: Military Simulation Performance Measurements

Delivery Ratio in Military Simulations

The delivery ratios for all protocols in the military scenario are lower then those in the urban and random mobility simulation. This is likely owing predominately to the geographical size of the area (over 60 km^2) and the clustered organization of the platoons. The clustered organization forces nodes to rely on data mules, namely the drones, Humvees, and helicopters, to forward all messages between platoons. There are no opportunities for direct contacts between platoons. The only way for messages to propagate outside of the local area is for them to be carried by a Humvee, or a Helicopter if the helicopter passes over the destination on the return trip to the ships (since helicopters only transit between the platoon area and the ships).

The delivery ratios for the military scenario are shown in Figure 4.16a. GAPR2 performed on par with GAPR and MaxProp, which in previous scenarios have consistently had higher delivery ratios then GAPR2. Additional simulations were conducted to isolate the different types of nodes that generate messages and determine which messages are causing the bottleneck in delivery ratio, and to determine why GAPR2 has a higher relative delivery ratio compared to GAPR and MaxProp when compared to previous simulations.

Additional simulations of GAPR2 and GAPR were run for each message generator individually to determine the origin of the messages that are not able to be delivered. When messages are only generated by the marine nodes, delivery ratio of GAPR2 is 11.32 %. However, when messages are generated only by the Humvees the GAPR2 delivery ratio is 88.22 %, and when the messages are only generated on the ships 100 % of messages are delivered. Since the bottleneck in delivery is between the platoons, GAPR was simulated with only the marine node message generator as well. The delivery ratio of GAPR for messages generated by the marines is only 11.02 %, 2.7 % lower then the delivery ratio of GAPR2.

Overhead Ratio in Military Simulations

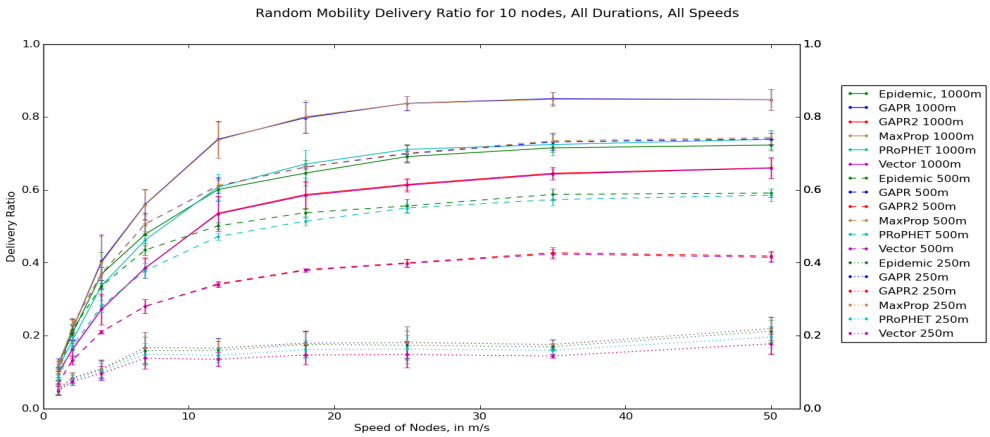
The overhead ratios for the military scenario are shown in Figure 4.16b. As in previous simulations, the overhead ratio of GAPR2 and Vector are significantly lower then all other protocols. Epidemic is not included, as the overhead ratio of Epidemic was many times larger then the other protocols in the graph. The overhead data from the additional simulations above for GAPR2 and GAPR were also analyzed to see what is happening in the marine clusters when only the marine message generator is active.

In this scenario the marine nodes generated 10,788 messages. Under GAPR2, 24,588 messages were forwarded to other marine nodes, while GAPR causes 97,885 messages to be forwarded, of which over 96,000 were dropped. GAPR incurred nearly four times the overhead of GAPR2 in marine-to-marine message propagation. There are so many dependent variables at play here that it is difficult to attribute the delivery ratio effects described in the previous section to the overhead incurred by GAPR discussed above or the latency effects discussed next. However, it is clear constrained resources effected GAPR2 less then GAPR in the marine clusters.[16]

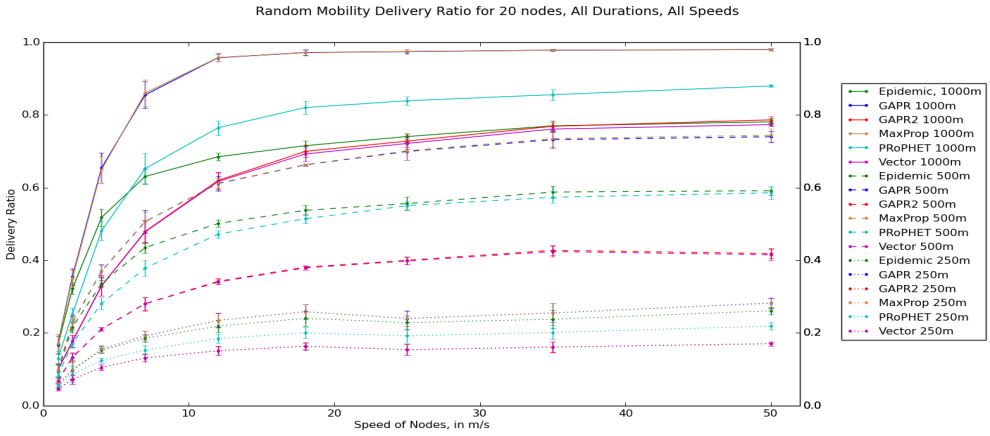
Latency in Military Simulations

The latency incurred by each protocol in the military simulation is shown in Figure 4.16c. Similar trends to those established previously are noted; namely, while GAPR2 does has higher latency then GAPR, the difference in latency between the two is less pronounced then in previous simulations. When the marine nodes are isolated as in previous sections, the latency between marine nodes is higher under GAPR then GAPR2, 9,739.1 seconds to 9,695.0 seconds. This is once again a reversal of the norms established in previous scenarios.

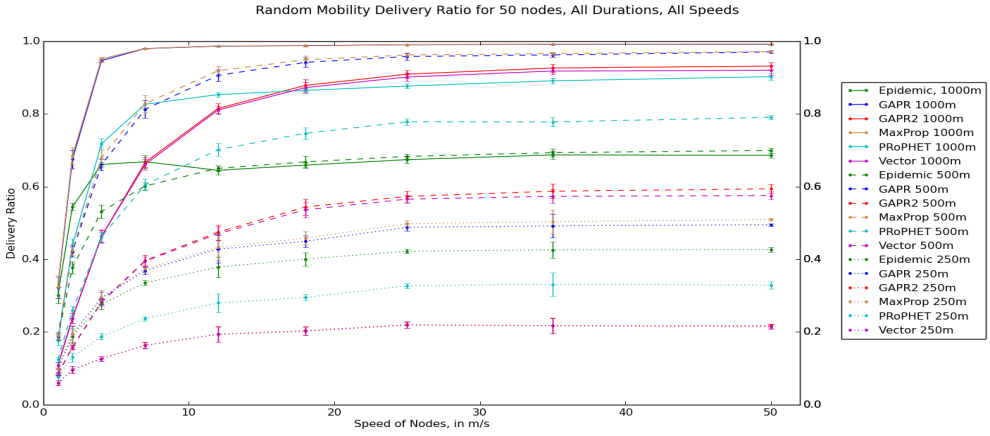
[16] Additional simulation and other approaches are possible; see the research methods and future work sections in Chapter 5.



(a) Delivery Ratio of 10 Node Networks

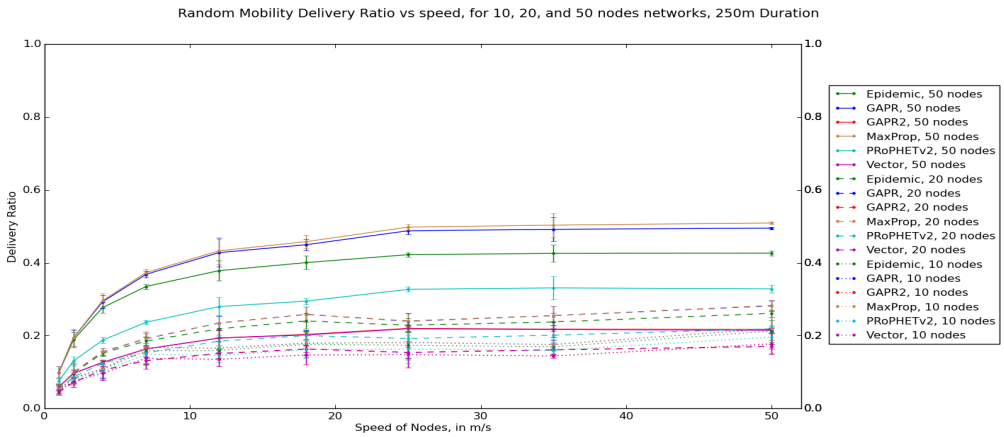


(b) Delivery Ratio of 20 Node Networks

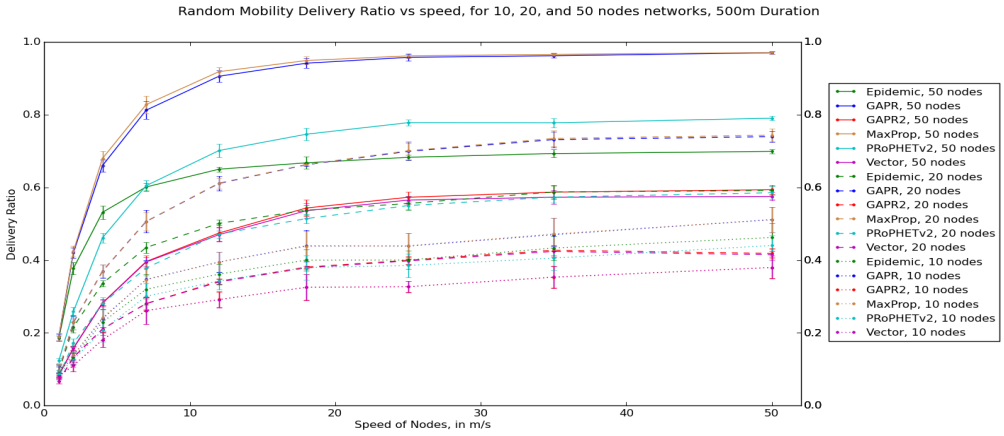


(c) Delivery Ratio of 50 Node Networks

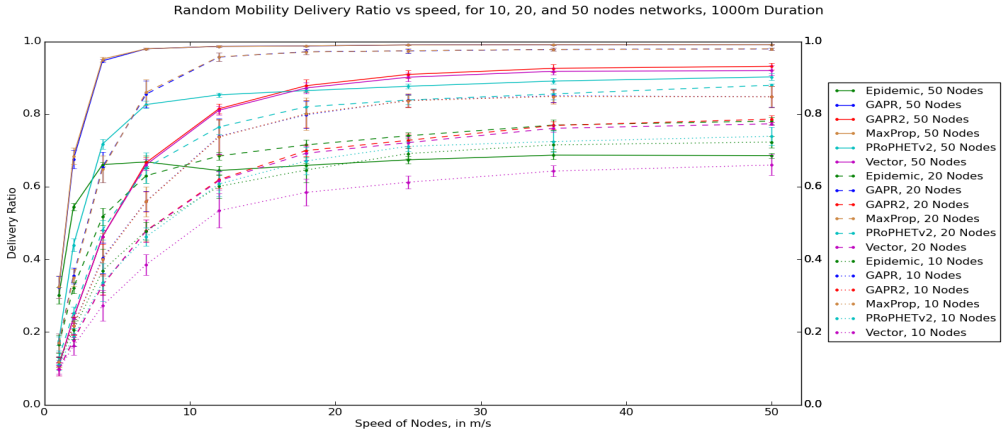
Figure 4.11: Delivery Ratio of Random Mobility Simulations for Specified Network Size, Duration as Series



(a) Delivery Ratio with 250 m Duration



(b) Delivery Ratio with 500 m duration



(c) Delivery Ratio with 1000 m duration

Figure 4.12: Delivery Ratio of Random Mobility Simulations for Specified Duration, Network Size as Series

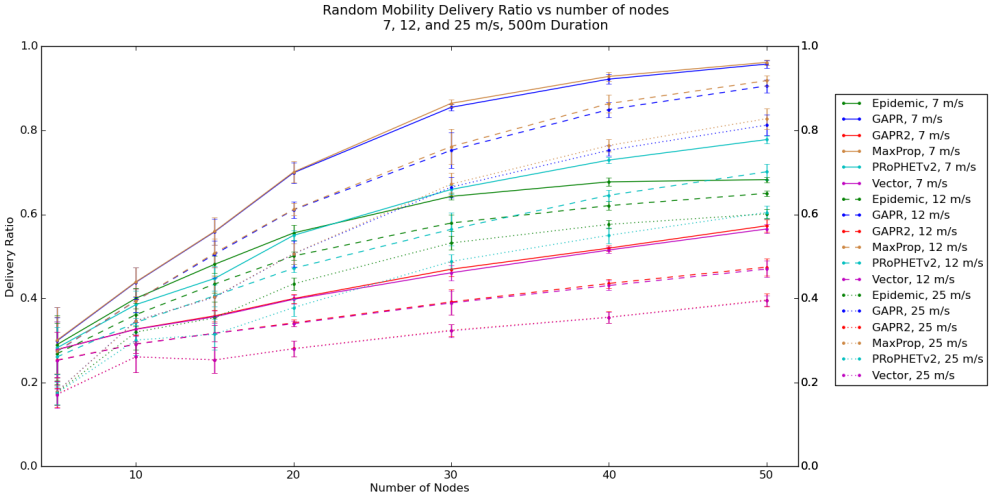
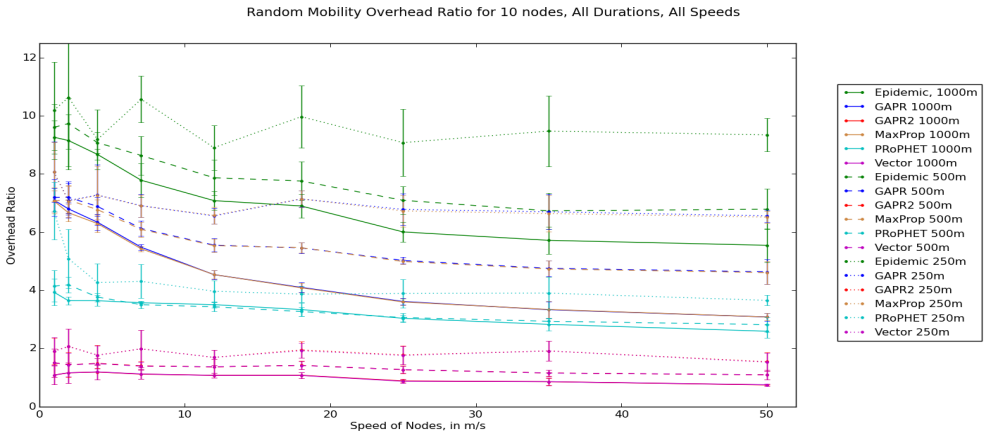
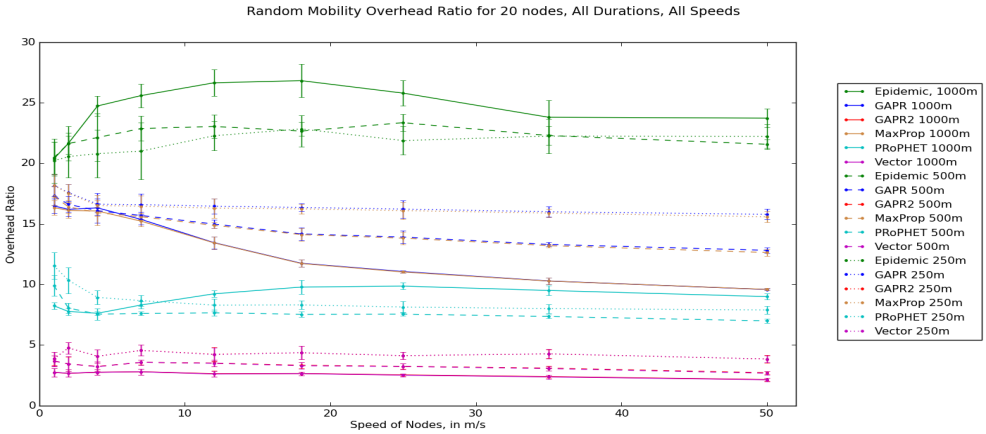


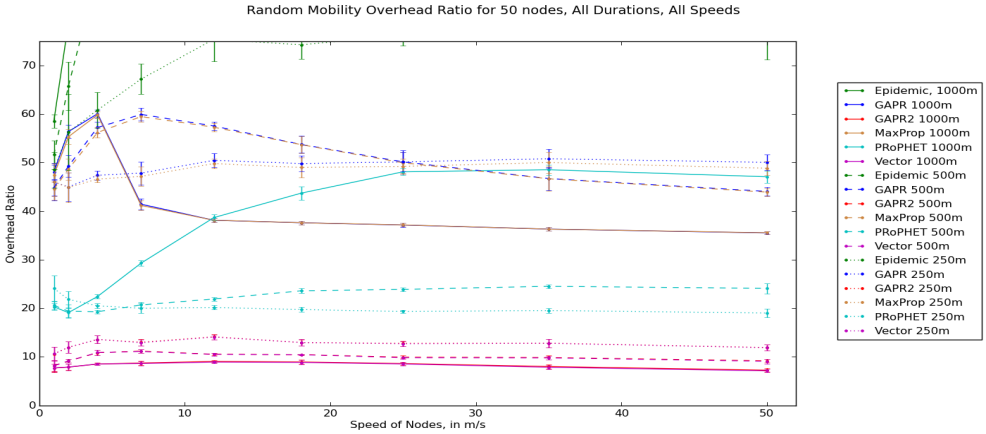
Figure 4.13: Random Mobility Delivery Ratio, 500 m duration



(a) Overhead Ratio for 10 Node Networks

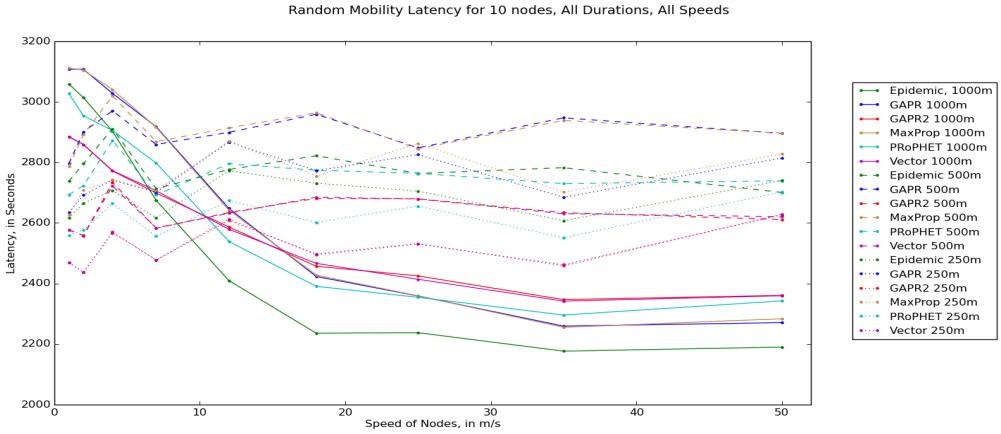


(b) Overhead Ratio for 20 Node Networks

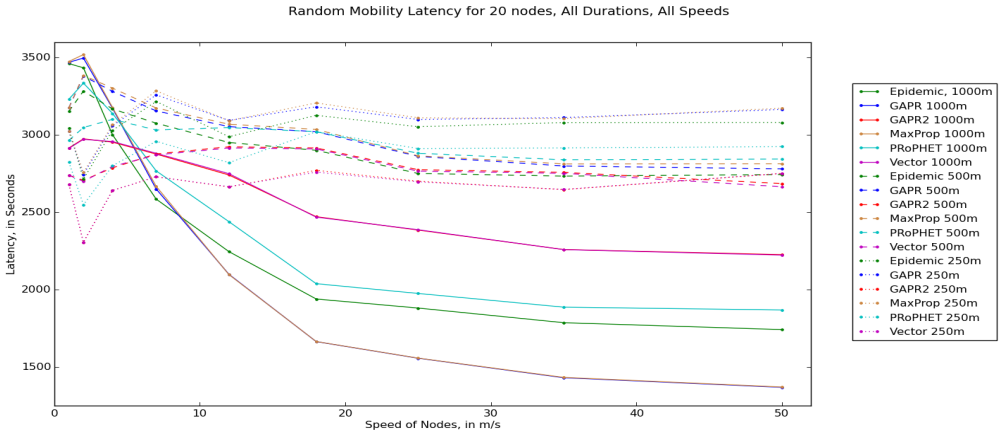


(c) Overhead Ratio for 50 Node Networks

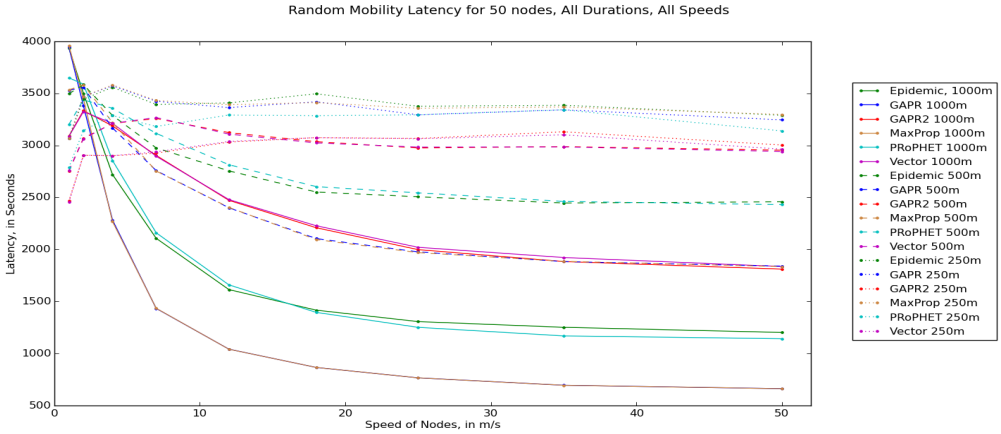
Figure 4.14: Overhead Ratio of Random Mobility Simulations



(a) Latency for 10 Node Networks

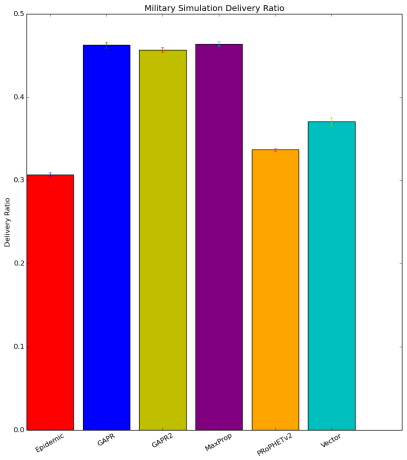


(b) Latency for 20 Node Networks

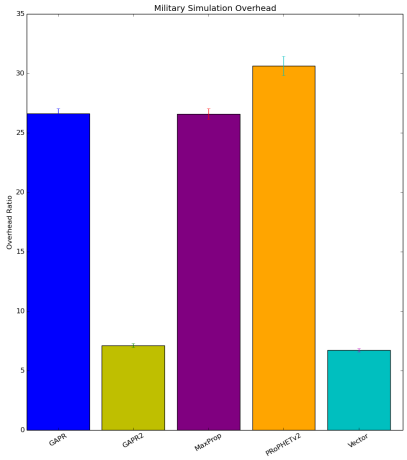


(c) Latency for 50 Node Networks

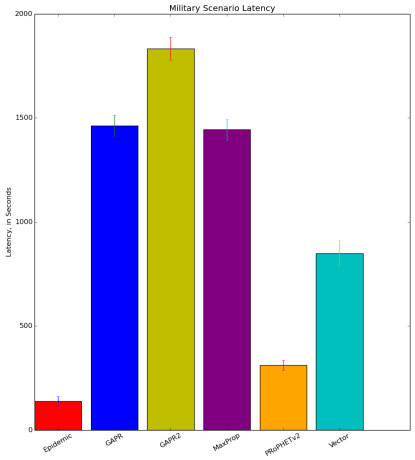
Figure 4.15: Latency of Random Mobility Simulations



(a) Delivery Ratio



(b) Overhead Ratio



(c) Latencey

Figure 4.16: Performance Metrics of Protocols in Military Simulation

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusions

This thesis explores the development of DTN routing protocols, from their origins in precursor technologies to the underlying strategies that inform modern DTNs. It outlines how the top performing protocols in the field operate in a comprehensible yet detailed manner. This work then introduces a high-performing protocol developed by Rohrer et. al. [2] which was never published, implements the Vector protocol [1] in ONE, and combines the two into a new protocol, GAPR2. All of these protocols are simulated in ONE over three mobility scenarios.

5.1 Approach to Simulation

This thesis models three mobility scenarios, the Helsinki scenario, a random mobility scenario, and a military scenario. The Helsinki Scenario was developed by the same group of researchers that developed The ONE simulator, and is used often in other research. The Helsinki Scenario provides a common link to other research in the field and allows the findings from this work to be directly compared to future research. Further, the Helsinki Scenario provides a realistic urban environment different from the other scenarios in this work. The random mobility scenario provides the opportunity to evaluate protocol performance when contact schedules cannot be predicted, and is the only scenario that doesn't severely limit node movement by forcing nodes to remain on paths. The military scenario created for this thesis provides a realistic rural topography built using over sixty square kilometers of actual GIS data and a set of node attributes derived from the characteristics of military equipment. This scenario and derivatives like it can be used to evaluate DTN protocol performance in military Humanitarian Assistance / Disaster Relief (HADR) operations or large-scale evacuations.

This thesis also provides a description of the process used to create the military scenario. The ONE simulator is widely used by DTN researchers, and the majority of the questions regarding its use[17] relate to creating customized scenarios from real-world topography. QGIS is a excellent tool and our hope is that many more researchers can contribute complex large-scale scenarios based on real-world topography using this development process.

[17] As observed via The ONE mailing list

5.2 Findings

MaxProp and GAPR provide excellent delivery probability and low latency in urban environments as modeled by the Helsinki simulations, however these protocols both incur significant overhead. Asymptotic growth in delivery ratio is reached by all of the top performing protocols including GAPR, GAPR2, Vector, and MaxProp when transmission speed is greater then 0.5 MBps, which equates to the ability to transmit 2 messages every three seconds on average, and a buffer capable of storing more then 5-7 messages. Provisioning additional resources to nodes yields marginal improvements to performance.

When provided with sufficient resources, GAPR and MaxProp yield a delivery ratio approximately 10% higher then GAPR2 and Vector with half to a quarter of the latency. However, overhead incurred by GAPR and MaxProp is between five and eight times larger then that of GAPR2 and Vector. The performance metrics of GAPR and MaxProp closely resemble one another in the Helsinki Scenario, as do the performance metrics of GAPR2 and Vector. Since overhead equates to additional transmission and computation cycles, a rough estimate of relative power consumption can be made, and if power consumption is a primary concern GAPR2 would be recommended in mobile DTNs with the understanding that delivery ratio will likely decrease.

It is expected that knowledge-based protocols would perform worse as compared to protocols that less dependent on knowledge in a random mobility scenario. Simulation in this thesis does not support that assumption; MaxProp and GAPR, both of which rely heavily on gaining knowledge of a network to inform routing decisions, continue to perform better significantly better then both Epidemic and Vector. Epidemic, the least knowledge-dependent protocol simulated, does perform better relative to is previous performance as compared to MaxProp and GAPR. However, Epidemic still averages 30% lower delivery ratio under unconstrained simulations and incurs several times larger overhead and significantly higher latency then MaxProp and GAPR. MaxProp, GAPR, and GAPR2 employ buffer optimizations, mechanisms to order and promote messages according to the average number of bytes transferred during previous encounters, and other adaptive mechanisms that complicate positive attribution of performance in the random mobility scenarios to a single factor.

The military simulation built for this thesis fundamentally represents a clustered network with data mules that transit between clusters delivering messages. Within each marine cluster node density is high and resources (buffer size and bandwidth) are scarce. The military simulations

indicate that in such an environment limiting replication can not only decrease overhead as expected, but can actually improve delivery ratio and reduce latency. The vector mechanism employed by GAPR2 must be responsible for the improvement in the performance of GAPR2 in this scenario, as it is the only difference between GAPR and GAPR2. However, the underlying cause, be it buffer exhaustion, overhead collisions, or another factor, is difficult to ascertain due to the number of competing dependent variables.

5.2.1 Summary

GAPR outperforms MaxProp in simulation when transmission speed is very high or when resources are very constrained. GAPR2 consistently outperforms Vector, and significantly reduces overhead while maintaining a higher delivery ratio then Epidemic and Vector. Generally, delivery ratio and latency are improved at the cost of increased overhead. However, this generalization breaks down under the most constrained environments. Therefore, GAPR is shown a excellent protocol for networks that are not overhead-limited or power aware, in which cases GAPR2 may be more appropriate.

This thesis shows that the framework of the scenario, be it urban, random, or military-oriented, greatly effects to performance of all protocols. Many researchers point out how uniquely well-suited DTNs are for use during military and disaster relief operations, but no research we reviewed actually includes simulation based on such operations. This thesis also shows that combining the logic of different protocols to leverage their most effective mechanisms can improve the overall performance of the derived protocol.

5.3 Recommendations for Future Work

The field of DTN research remains a active and open area, and as with most research in the field, our research and the conclusions we have drawn also lead to more questions. Below are suggestions for future work for which this thesis can be used as a starting point.

We explained the precepts of Vector Routing and implemented Vector in ONE. We also combined Vector with GAPR to form GAPR2. We did not aim specifically to customize Vector for optimal performance. The algorithm to determine the current heading of a node running Vector, equation 3.1, relies of a input of past movement, Θ. This is predicated on the use of a weighted average of movement history. Simulation can be used to determine the optimal formula for calculation of the weighted average.

GAPR2 is a interesting protocol that would benefit from additional simulation and further refinement. GAPR2 significantly reduces overhead which should correspond to significant power savings. Repeating or expanding on the simulation in this work and incorporating a power model to track battery consumption should confirm that GAPR2 is appropriate for power-aware networks and provide data on the power consumption of the protocols simulated. In the last section of chapter 4 we noted that under the most resource-scarce conditions GAPR2 performs particularly well compared to other protocols. We were unable to conduct the additional simulations that are required to investigate this completely. Further simulation of GAPR2 in highly resource-constrained environments with an emphasis on determining causation could provide insight for future protocols. Finally, the Vector logic in GAPR2 could be implemented in such a way that it is only activated when buffers are full. This would allow increased replication in unconstrained environments, increasing delivery ratio and reducing latency.

This work describes a process for designing large-scale customized DTN simulations using Quantum GIS. Currently researchers using ONE favor the OpenJump tool and API exports from OSM. However, the Quantum GIS tool used in this work provides support for importing arbitrary-sized OSM maps directly as a base layer, for the creation of overlays that can be exported directly into WKT format, and a host of layer editing and customization options. QGIS can be used to create additional complex, large-scale simulations based on real GIS data.

This work leveraged QGIS to create a large custom mobility model based on a military exercise. However, it should not be difficult to collect actual track data during the next Bold Alligator simulation, or to obtain past track data if available. This track data, stored as a CSV of points, can be imported to QGIS and used for simulation based on actual historical unit movements to evaluate how a DTN network would have performed had it been employed. Working in conjunction with DARPA, such information could be relevant to ongoing efforts related to the WNAN discussed in Chapter 2, or could be used to guide the development of future protocols.

In our literature review we noted that many researchers believe protocols must be customized for the environment in which they'll be employed. We have shown that current protocols can be used as building blocks, and that future protocols can be improved by incorporating the best mechanisms found within current and future protocols. By learning about the network and activating or deactivating those mechanisms, future protocols should adapt to changing network conditions. We concur with other researchers that current protocols are largely designed with a target environment and performance metric in mind. Future protocol development should be

aimed at creating adaptive DTN protocols that can activate and deactivate mechanisms based on the state of the node and learned knowledge of the network could provide significant performance improvements.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A:
Helsinki Simulation Set One Aggregate Data

This appendix includes the means and confidence intervals of the performance measurements for the first set of Helsinki Simulations.

Graph 1, Buffer Size vs Deliver Ratio for 250kBps, fast message generator						
Buffer Size	4	6	9	14	22	35
Epidemic Delivery Ratio	0.11486	0.16252	0.22776	0.31104	0.42788	0.55082
95% CI	0.00265234	0.00282243	0.00442109	0.00289284	0.00611663	0.01039533
Gapr2 Delivery Ratio	0.1732	0.25854	0.36978	0.45666	0.47472	0.4732
95% CI	0.00455794	0.0062637	0.00898402	0.01475246	0.00996601	0.01179646
Gapr Delivery Ratio	0.17276	0.25946	0.38658	0.53838	0.68498	0.75998
95% CI	0.00555816	0.00848911	0.00792888	0.010901	0.01463042	0.01235047
MaxProp Delivery Ratio	0.17284	0.26136	0.39144	0.54614	0.68416	0.76152
95% CI	0.00491093	0.01020154	0.01097823	0.01318639	0.0135619	0.0178304
Prophet Delivery Ratio	0.15124	0.20742	0.27128	0.3465	0.44734	0.5496
95% CI	0.00428203	0.00444179	0.00712817	0.00838008	0.01011465	0.01663482
Vector Delivery Ratio	0.1419	0.20766	0.30516	0.40836	0.44616	0.4479
95% CI	0.00634951	0.00313087	0.00488181	0.01397151	0.01416439	0.01291957

Graph 2, Buffer Size vs Overhead Ratio for 250kBps, fast message generator						
Buffer Size	4	6	9	14	22	35
Epidemic Overhead	65.20644	48.17726	35.25012	25.8019	18.18884	13.50198
95% CI	1.73799549	1.72931342	1.27281172	0.93677691	0.18451788	0.12262409
Gapr2 Overhead	5.45638	4.39388	3.59286	2.99376	2.89114	2.90076
95% CI	0.13467466	0.13449283	0.05862118	0.04976832	0.06452524	0.06667733
Gapr Overhead	32.91326	23.21822	16.14312	11.79736	9.33352	8.4136
95% CI	0.67211111	0.29425461	0.27973669	0.21114509	0.09382681	0.10786548
MaxProp Overhead	32.90936	23.02798	15.93746	11.54548	9.29132	8.37712
95% CI	0.78446076	0.53210659	0.29681438	0.18406318	0.11451866	0.12085363
Prophet Overhead	31.1271	25.67518	21.03754	16.98358	13.40536	10.87046
95% CI	0.86720475	0.31456455	0.19579456	0.22225807	0.24251895	0.11361943
Vector Overhead	5.67372	4.8122	3.98898	3.17306	3.00264	2.99008
95% CI	0.22517582	0.15173855	0.11702765	0.04652552	0.05656966	0.03572093

Graph 3, Buffer Size vs Latency (in seconds) for 250kBps, fast message generator						
Buffer Size	4	6	9	14	22	35
Epidemic Latency	2098.2	2600.52	3386.96	4462.22	6121.34	7703.18
95% CI	60.430015	30.4184112	184.792323	92.8986002	149.066859	236.091649
Gapr2 Latency	4776.9	5834.5	6688.22	7227.44	7286.86	7270.68
95% CI	199.472036	182.54093	165.939937	162.708462	146.748259	185.785475
Gapr Latency	5123.46	5829.06	6485.58	6478.48	6409.5	6424.06
95% CI	146.472561	206.702312	186.997162	165.17355	46.8695678	70.9902103
MaxProp Latency	5059.02	5834.16	6567.46	6688.74	6555.58	6486.4
95% CI	237.086294	196.659011	130.223403	193.92873	77.3861247	68.6715966
Prophet Latency	2104.44	2608.26	3235.22	4074.58	5540.58	7290.4
95% CI	103.954777	51.8787219	65.2933119	135.313447	176.614459	127.005324
Vector Latency	3895.16	4834.78	5887.32	6881.3	7222.62	7250.04
95% CI	219.347204	222.14749	134.627255	132.078327	88.1075184	127.263673

Graph 4, Buffer Size vs Deliver Ratio for 2MBps, fast message generator						
Buffer Size	4	6	9	14	22	35
Epidemic Delivery Ratio	0.1401	0.1839	0.23616	0.29206	0.35878	0.44424
95% CI	0.00550899	0.00441446	0.00407447	0.00673916	0.00781923	0.00436494
Gapr2 Delivery Ratio	0.40468	0.5905	0.75774	0.83102	0.85298	0.85622
95% CI	0.00825142	0.00954831	0.01351721	0.00828265	0.00899517	0.00752893
Gapr Delivery Ratio	0.43458	0.65366	0.84406	0.95536	0.97644	0.98126
95% CI	0.00696019	0.00835964	0.01299056	0.00401636	0.0025319	0.0018492
MaxProp Delivery Ratio	0.4122	0.60914	0.80434	0.95424	0.97692	0.9815
95% CI	0.00868499	0.01182048	0.0085655	0.00410088	0.00295062	0.00211448
Prophet Delivery Ratio	0.24168	0.30892	0.3747	0.4461	0.51836	0.59044
95% CI	0.00434618	0.0029947	0.00606067	0.00379064	0.00651998	0.00837208
Vector Delivery Ratio	0.24956	0.35188	0.49796	0.6571	0.78782	0.81812
95% CI	0.00446101	0.00844852	0.01813947	0.01092453	0.00996794	0.00741649

Graph 5, Buffer Size vs Overhead Ratio for 2MBps, fast message generator						
Buffer Size	4	6	9	14	22	35
Epidemic Overhead	220.14404	224.16024	210.66604	198.14546	181.95636	159.05452
95% CI	12.1556476	5.53127277	3.70535577	4.50861429	2.54238059	0.98230072
Gapr2 Overhead	9.55578	9.7367	10.1082	9.57502	9.20206	9.15732
95% CI	0.40244358	0.23659806	0.19541375	0.21279313	0.21091458	0.1837727
Gapr Overhead	83.72482	68.06372	63.07808	63.36258	64.70488	64.00538
95% CI	1.41020594	0.75722295	0.71486865	0.62250826	0.75507035	0.58518906
MaxProp Overhead	90.8404	75.4035	66.64606	63.44802	64.83984	64.18514
95% CI	1.35098535	1.26088235	0.91832354	0.78478046	0.55128537	0.65095688
Prophet Overhead	87.03368	87.64872	88.4714	87.59364	85.74776	84.57358
95% CI	1.62983366	1.03311024	1.00993916	1.08266651	0.98378207	1.69464003
Vector Overhead	9.86216	9.82398	9.68478	9.0182	8.90208	8.93178
95% CI	0.20023415	0.22540902	0.19030159	0.17083941	0.24535204	0.24022836

Graph 6, Buffer Size vs Latency (in seconds) for 2MBps, fast message generator						
Buffer Size	4	6	9	14	22	35
Epidemic Latency	1281.9	1426.62	1625.2	1827.18	2073.26	2341.16
95% CI	53.2463252	90.2982333	62.994413	107.519693	115.416713	97.6665071
Gapr2 Latency	3350.58	3928.34	3970.42	3852.72	3826.88	3830.4
95% CI	79.6813335	59.1139395	72.4794917	54.0140537	56.3290969	87.0943879
Gapr Latency	2821.22	3507.06	2888.06	1956.54	1466.76	1246.6
95% CI	68.456935	98.7067275	123.029468	34.5284857	37.0770933	38.8857963
MaxProp Latency	1583.78	2078.3	2314.84	1906.3	1447.54	1236.38
95% CI	44.333004	94.6755161	69.0585454	38.5237136	37.4763116	37.1591057
Prophet Latency	1482.92	1737.26	1914.92	2051	2148.16	2205.9
95% CI	51.8415463	32.4873264	60.9248098	30.9128206	43.7845129	46.0965983
Vector Latency	2161.24	2559.98	3159.52	3645.18	3999.9	4070.96
95% CI	93.2827959	70.2545077	70.7067632	129.221745	41.1480967	73.8408403

Graph 7, Buffer Size vs Deliver Ratio for 250kBps, slow message generator						
Buffer Size	4	6	9	14	22	35
Epidemic Delivery Ratio	0.16822	0.24056	0.32702	0.44324	0.59248	0.68636
95% CI	0.00688614	0.00886404	0.00388444	0.01220582	0.00812669	0.00773558
Gapr2 Delivery Ratio	0.36454	0.48638	0.54462	0.56874	0.566	0.56618
95% CI	0.00699455	0.00559121	0.00958111	0.00913725	0.00798632	0.00882211
Gapr Delivery Ratio	0.38456	0.5784	0.7863	0.90352	0.92528	0.92794
95% CI	0.00945607	0.01236781	0.01918602	0.0075606	0.0054523	0.00505023
MaxProp Delivery Ratio	0.39374	0.58214	0.78974	0.90518	0.9249	0.92742
95% CI	0.00458542	0.0038317	0.01503455	0.00571057	0.00466905	0.00482607
Prophet Delivery Ratio	0.24928	0.32066	0.3985	0.5011	0.61376	0.70202
95% CI	0.00327385	0.00453981	0.00205157	0.00534276	0.0062118	0.00386056
Vector Delivery Ratio	0.29076	0.40266	0.48802	0.52242	0.52404	0.52404
95% CI	0.0112973	0.02090805	0.00958674	0.00998108	0.00920826	0.00920826

Graph 8, Buffer Size vs Overhead Ratio for 250kBps, slow message generator						
Buffer Size	4	6	9	14	22	35
Epidemic Overhead	114.83254	83.21464	62.41294	44.82978	32.68394	27.19504
95% CI	3.25611389	2.72246237	2.06364284	1.33118137	0.98766897	0.47204748
Gapr2 Overhead	6.12344	5.33128	5.09276	5.07628	5.06414	5.09944
95% CI	0.19093958	0.16840366	0.15360444	0.14361205	0.18906873	0.15571424
Gapr Overhead	37.3725	26.41684	20.28542	18.26894	18.08876	18.10318
95% CI	0.82311762	0.38615419	0.43917295	0.18673476	0.13026797	0.11742119
MaxProp Overhead	36.45976	26.18554	20.09756	18.18962	18.04538	18.07032
95% CI	0.72751393	0.30181641	0.38038422	0.13707392	0.11794626	0.13478085
Prophet Overhead	44.99192	39.85038	34.79706	28.8647	24.2519	21.24678
95% CI	2.09078586	1.51661787	0.9405017	0.83627679	0.4460327	0.34639081
Vector Overhead	5.9141	5.51702	5.24864	5.21516	5.2055	5.2055
95% CI	0.2747575	0.36784136	0.13791331	0.07663122	0.13209712	0.13209712

Graph 9, Buffer Size vs Latency (in seconds) for 250kBps, slow message generator						
Buffer Size	4	6	9	14	22	35
Epidemic Latency	2767.92	3298.4	4140.6	5393.8	6921.04	7844.22
95% CI	98.8091682	185.243785	155.515206	180.354634	320.261283	155.717671
Gapr2 Latency	6194.22	6862.78	7005.34	7100.9	7043.46	7076.68
95% CI	345.262971	348.185516	286.594888	190.388921	270.239143	216.532948
Gapr Latency	5840.82	5925.8	5509.12	4847.92	4586.98	4564.82
95% CI	210.897414	279.504247	153.53401	104.722504	127.526642	126.009517
MaxProp Latency	5913.92	6038.68	5603.74	4864.2	4609.88	4581.62
95% CI	269.454768	169.927364	201.852955	236.777283	114.230368	141.287544
Prophet Latency	2867.34	3329.94	3923.76	5009.08	6169.02	7255.2
95% CI	127.484751	84.8615166	162.572718	143.295299	267.111384	269.665624
Vector Latency	5380	6334.3	6802.86	7059.14	7081.88	7081.88
95% CI	155.17698	363.636539	366.603252	360.454092	283.632334	283.632334

Graph 10, Buffer Size vs Delivery Ratio for 2MBps, slow message generator						
Buffer Size	4	6	9	14	22	35
Epidemic Delivery Ratio	0.17752	0.23838	0.2971	0.37368	0.43752	0.52176
95% CI	0.01176387	0.01148435	0.01179156	0.01556776	0.01080904	0.00566599
Gapr2 Delivery Ratio	0.71292	0.84376	0.88834	0.88998	0.8949	0.89514
95% CI	0.01186988	0.00753619	0.01090602	0.01014615	0.01097065	0.01001316
Gapr Delivery Ratio	0.82098	0.9561	0.9795	0.98488	0.98692	0.98708
95% CI	0.01569745	0.00497828	0.00257926	0.00203345	0.00173922	0.00178299
MaxProp Delivery Ratio	0.72888	0.9227	0.97956	0.9851	0.98692	0.98732
95% CI	0.01380103	0.0070513	0.00173077	0.00224875	0.00211521	0.00173922
Prophet Delivery Ratio	0.34518	0.40954	0.46538	0.526	0.59202	0.6661
95% CI	0.00546572	0.01091062	0.01134252	0.01046687	0.01106628	0.01143511
Vector Delivery Ratio	0.48018	0.63266	0.77892	0.85842	0.88056	0.879
95% CI	0.0119669	0.01108376	0.01241894	0.01049166	0.01098701	0.01046944

Graph 11, Buffer Size vs Overhead Ratio for 2MBps, slow message generator						
Buffer Size	4	6	9	14	22	35
Epidemic Overhead	420.27404	418.31244	406.84376	378.87678	369.68308	334.40896
95% CI	19.7405647	21.8643954	11.6116705	13.44947	15.6329492	9.60250437
Gapr2 Overhead	11.30984	11.8538	12.94226	12.29646	12.91784	12.82914
95% CI	0.42653754	0.30977582	0.56929108	0.39754181	0.48097463	0.53490132
Gapr Overhead	112.41962	116.48822	123.61776	119.30046	89.55506	87.25638
95% CI	2.3040646	1.22002289	1.18647598	1.39100525	1.08712443	0.93610643
MaxProp Overhead	125.1667	119.88858	123.99496	119.91834	89.78928	87.40244
95% CI	2.56772319	0.45401815	1.67337349	1.86632156	0.99892567	0.95103614
Prophet Overhead	140.40636	154.08856	168.10422	176.92672	180.1424	179.28572
95% CI	4.02208354	5.03219713	6.22966782	4.63502082	4.63791488	2.20057164
Vector Overhead	10.16734	10.89806	11.91094	11.93222	12.51828	12.56414
95% CI	0.30279778	0.22597078	0.50442257	0.29906144	0.51377595	0.34096673

Graph 12, Buffer Size vs Latency (in seconds) for 2MBps, slow message generator						
Buffer Size	4	6	9	14	22	35
Epidemic Latency	1740.3	1815.52	1915.08	2105.98	2198.26	2412.26
95% CI	65.4270924	53.5978615	133.269521	41.6541168	115.935272	20.507313
Gapr2 Latency	3846.56	3867.9	3656.58	3703.72	3636.58	3640.46
95% CI	37.2163385	72.6141519	122.240998	83.2644453	121.066105	140.035524
Gapr Latency	2520.96	1886.4	1365.56	1068.26	903.68	897.2
95% CI	133.186671	72.3899695	27.3813799	28.2680565	26.8649437	26.1659882
MaxProp Latency	1590.06	1625.76	1328.16	1053.96	902.8	898.5
95% CI	33.6482467	42.3458944	25.0580285	31.5383389	28.388659	26.705592
Prophet Latency	2116.82	2217.08	2269	2334.54	2275.12	2209.08
95% CI	63.3021203	84.9516618	92.1417177	56.2723576	85.6374408	48.8529225
Vector Latency	3090.58	3424.98	3708.66	3852.16	3800.88	3812.5
95% CI	121.822217	43.9967479	61.0281708	152.379103	74.2913314	56.7788794

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B:
Helsinki Simulation Set Two Aggregate Data

This appendix includes the means and confidence intervals of the performance measurements for the second set of Helsinki Simulations.

Graph 1, Transmit Speed vs Delivery Ratio for 10MB buffer, fast message generator						
Transmit Speeds	0.125	0.25	0.5	1	1.5	2
Epidemic Delivery Ratio	0.17224	0.24688	0.29116	0.27922	0.25836	0.24782
95% CI	0.00804911	0.00300883	0.00453301	0.00717022	0.00536407	0.00434795
Gapr2 Delivery Ratio	0.19328	0.39588	0.66684	0.75646	0.77234	0.77794
95% CI	0.00695853	0.01164235	0.01044128	0.00937954	0.01340554	0.01148341
Gapr Delivery Ratio	0.20236	0.42308	0.75494	0.85472	0.86898	0.874
95% CI	0.00685877	0.01160587	0.01316591	0.00925993	0.01049423	0.0096239
MaxProp Delivery Ratio	0.1995	0.42664	0.7736	0.86234	0.86176	0.84692
95% CI	0.00571907	0.01066372	0.01069022	0.00995486	0.00963814	0.00291117
Prophet Delivery Ratio	0.17	0.288	0.40062	0.41302	0.40074	0.39054
95% CI	0.00864763	0.00692833	0.009749	0.00591806	0.00456436	0.00393198
Vector Delivery Ratio	0.17668	0.33258	0.51716	0.55086	0.5403	0.53042
95% CI	0.00570112	0.00899431	0.00960449	0.00795809	0.009864	0.01142783

Graph 2, TransmitSpeed vs Overhead Ratio for 10MB buffer, fast message generator						
Transmit Speeds	0.125	0.25	0.5	1	1.5	2
Epidemic Overhead	17.02532	32.2508	62.73278	117.5104	168.24802	209.47584
95% CI	0.65446639	0.49124274	1.69437079	3.22643135	2.96611376	5.18550098
Gapr2 Overhead	2.74522	3.42278	4.55468	7.02522	8.75538	10.04028
95% CI	0.06525446	0.04628105	0.05785084	0.20730278	0.13614857	0.22233081
Gapr Overhead	12.11142	14.80244	20.60088	37.25672	51.54136	63.33058
95% CI	0.13554164	0.2492687	0.39219291	0.64795796	0.60315575	0.61869922
MaxProp Overhead	12.28046	14.62284	20.08868	36.99476	52.0273	65.6507
95% CI	0.08389458	0.23996169	0.34888802	0.3827011	0.90764045	0.70197862
Prophet Overhead	13.52652	19.98718	31.88366	54.00374	72.80844	88.32726
95% CI	0.62401931	0.42485949	0.60721664	0.75339939	0.51378021	1.52695716
Vector Overhead	2.89814	3.70234	4.93532	7.16638	8.38034	9.26204
95% CI	0.11717429	0.04208587	0.10742889	0.16276672	0.13938381	0.19570347

Graph 3, Transmit Speed vs Latency for 10MB buffer, fast message generator						
Transmit Speeds	0.125	0.25	0.5	1	1.5	2
Epidemic Latency	5458.18	3677.68	2175.84	1758.9	1676.88	1615.34
95% CI	156.267876	100.743088	61.4239521	52.6690246	62.3346492	57.382518
Gapr2 Latency	7398.08	6821.32	5026.36	4130.62	3990.96	3898.44
95% CI	204.942829	130.938882	107.235806	101.539741	91.2835702	84.6910697
Gapr Latency	7390.78	6567.34	4090.42	2899.82	2745.48	2640.72
95% CI	173.808089	224.979659	99.0583087	78.6504418	57.4171533	47.0451726
MaxProp Latency	7428.62	6678.14	4105.28	2853.08	2539.92	2253.88
95% CI	106.454257	146.746894	101.083438	102.274281	67.422378	47.0857095
Prophet Latency	4965.52	3397.44	2242.56	1954.92	1942.84	1950.42
95% CI	102.794085	100.055262	67.309562	50.3522998	39.6164374	34.1107949
Vector Latency	7318.34	6180.76	4385.1	3576.64	3383.08	3244.56
95% CI	318.281507	126.742259	152.999786	124.903054	68.4773699	81.6539997

Graph 4, Transmit Speed vs Delivery Ratio for 50MB buffer, fast message generator						
Transmit Speeds	0.125	0.25	0.5	1	1.5	2
Epidemic Delivery Ratio	0.31874	0.59386	0.69024	0.59282	0.5461	0.52232
95% CI	0.01168075	0.01236083	0.00544183	0.00557948	0.00487818	0.00373058
Gapr2 Delivery Ratio	0.2378	0.47326	0.72062	0.81628	0.84258	0.8576
95% CI	0.00912264	0.01334358	0.013362	0.00965764	0.00806097	0.00608162
Gapr Delivery Ratio	0.35644	0.7708	0.95774	0.9765	0.98056	0.98276
95% CI	0.00680688	0.01508686	0.00289284	0.00245994	0.00191876	0.00250435
MaxProp Delivery Ratio	0.35548	0.7722	0.95746	0.97648	0.9808	0.98274
95% CI	0.00612733	0.0160319	0.00249664	0.00262135	0.00178212	0.00213732
Prophet Delivery Ratio	0.29794	0.58438	0.70972	0.69362	0.66846	0.64862
95% CI	0.01255389	0.01763696	0.00930146	0.00842659	0.00643249	0.00804278
Vector Delivery Ratio	0.229	0.44762	0.67856	0.77314	0.80088	0.82138
95% CI	0.00893826	0.01433071	0.00904653	0.00766198	0.00694966	0.00803223

Graph 5, Transmit Speed vs Overhead Ratio for 50MB buffer, fast message generator						
Transmit Speeds	0.125	0.25	0.5	1	1.5	2
Epidemic Overhead	7.53902	11.6445	26.84904	65.85104	103.56016	138.9834
95% CI	0.16574204	0.1418404	0.20972377	0.91546272	1.51848004	2.86920303
Gapr2 Overhead	2.19558	2.88992	4.3357	6.73926	8.20924	9.25872
95% CI	0.07176299	0.06504313	0.04881216	0.13694998	0.20190072	0.23041678
Gapr Overhead	6.68042	8.30624	17.66202	35.376	49.61186	60.90892
95% CI	0.11410461	0.0867125	0.30150784	0.37204902	0.54652571	0.61293901
MaxProp Overhead	6.68964	8.2726	17.65616	35.38894	49.68434	61.10678
95% CI	0.10547271	0.11625135	0.29345691	0.36838504	0.51493717	0.61677825
Prophet Overhead	7.4879	10.14184	21.0891	43.52284	64.22466	83.07684
95% CI	0.24517071	0.15070573	0.36377439	0.81415568	0.8782709	1.35548164
Vector Overhead	2.24568	2.99396	4.42292	6.7252	8.06696	9.00822
95% CI	0.06566949	0.05101468	0.07107582	0.1811715	0.15280845	0.23977664

Graph 6, Transmit Speed vs Latency for 50MB buffer, fast message generator						
Transmit Speeds	0.125	0.25	0.5	1	1.5	2
Epidemic Latency	9571.12	8266.5	5103.34	3294.1	2838.06	2617.58
95% CI	103.295005	111.911335	116.002043	51.2753112	71.1777646	84.4160499
Gapr2 Latency	8178.14	7284.92	5012.82	4096.48	3906.24	3835.22
95% CI	221.574246	197.354171	93.6471268	76.6356891	50.1687267	58.7987492
Gapr Latency	8799.1	6472.14	2637.34	1557.2	1302.36	1162.26
95% CI	177.196655	39.0561656	34.851697	34.7199983	38.6934857	28.4446225
MaxProp Latency	8790.52	6483.08	2623.58	1559.48	1303.42	1163.1
95% CI	217.389365	28.1659922	42.3049884	29.1158161	39.5768981	34.2209671
Prophet Latency	9278.02	7883.24	4234.34	2769.2	2431.3	2269.94
95% CI	137.988048	160.553603	117.280032	60.3324141	47.7868872	33.2800235
Vector Latency	8062.42	7229.58	5205.48	4353.22	4152.9	4049.44
95% CI	224.058198	87.8293818	115.710841	62.4586242	58.7276226	75.4171933

Graph 7, Transmit Speed vs Delivery Ratio for 10MB buffer, slow message generator						
Transmit Speeds	0.125	0.25	0.5	1	1.5	2
Epidemic Delivery Ratio	0.29084	0.35128	0.38144	0.35836	0.33834	0.3198
95% CI	0.01103776	0.00699334	0.00477161	0.0048826	0.00866527	0.00933398
Gapr2 Delivery Ratio	0.28724	0.55708	0.8011	0.87	0.88136	0.88778
95% CI	0.00697855	0.0050577	0.00585758	0.01171548	0.01052035	0.01008327
Gapr Delivery Ratio	0.4223	0.82186	0.9686	0.97682	0.9794	0.97972
95% CI	0.01830665	0.01284015	0.00569814	0.00323715	0.00293175	0.00148326
MaxProp Delivery Ratio	0.40022	0.82758	0.96938	0.9783	0.97988	0.9805
95% CI	0.01669805	0.01044453	0.00449613	0.00406823	0.00153435	0.00301471
Prophet Delivery Ratio	0.29804	0.4223	0.51072	0.50698	0.49044	0.48176
95% CI	0.00306744	0.00769531	0.00671383	0.00901614	0.01307337	0.00904908
Vector Delivery Ratio	0.26968	0.50946	0.73916	0.8061	0.80588	0.80994
95% CI	0.00951693	0.01050385	0.00734505	0.00765665	0.01127708	0.01512146

Graph 8, Transmit Speed vs Overhead Ratio for 10MB buffer, slow message generator						
Transmit Speeds	0.125	0.25	0.5	1	1.5	2
Epidemic Overhead	24.47404	56.95424	119.45682	227.82962	312.19158	394.89838
95% CI	1.19295822	1.72617981	2.38907239	6.89521337	8.91102509	12.4399485
Gapr2 Overhead	4.26212	5.13902	7.16408	9.95114	11.4123	12.55636
95% CI	0.10059775	0.17455114	0.26240002	0.30155198	0.30961225	0.32016996
Gapr Overhead	15.0427	19.64236	41.0173	77.31284	103.18524	124.5726
95% CI	0.75392882	0.22607608	0.50404056	0.86602578	1.16951171	1.68831451
MaxProp Overhead	15.78666	19.45026	40.92884	77.16316	103.4097	124.4458
95% CI	0.81329154	0.27144618	0.57848715	0.94399273	1.17403777	1.41950102
Prophet Overhead	18.67184	33.28338	60.05648	104.48374	141.0794	169.94654
95% CI	0.48454052	1.18506498	1.16819369	2.41224667	4.38293258	4.93342578
Vector Overhead	4.39678	5.21076	7.10976	9.71442	11.05074	11.86328
95% CI	0.12324109	0.18260611	0.1865783	0.28938817	0.3124079	0.33413413

Graph 9, Transmit Speed vs Latency for 10MB buffer, slow message generator						
Transmit Speeds	0.125	0.25	0.5	1	1.5	2
Epidemic Latency	7716.1	4560.9	2541.68	2077.72	2002.12	2023.76
95% CI	289.467978	159.666621	136.245437	157.936425	57.2846231	123.711571
Gapr2 Latency	8490.14	7048.42	4898.8	3901.52	3750.72	3679.26
95% CI	396.144124	242.088316	93.7894499	89.4284047	112.640874	86.6377494
Gapr Latency	8167.44	5349.9	2043.76	1421.22	1328.7	1278.98
95% CI	395.525909	157.657618	46.34257	47.0562316	19.423213	28.676982
MaxProp Latency	8371.62	5386.88	2019.38	1407.34	1308.12	1254.4
95% CI	376.511676	236.854485	51.5489038	48.4379157	26.5887784	29.3269366
Prophet Latency	7230.64	4220	2491.08	2232.76	2248.5	2292.2
95% CI	255.96122	210.142764	57.8821487	65.1435085	33.7875232	70.5415971
Vector Latency	8320.04	6900.12	4953.7	4062.84	3862.28	3757.32
95% CI	405.772865	245.950844	257.028212	184.64096	149.688927	88.3525472

Graph 10, Transmit Speed vs Delivery Ratio for 50MB buffer, slow message generator						
Transmit Speeds	0.125	0.25	0.5	1	1.5	2
Epidemic Delivery Ratio	0.43978	0.72788	0.7891	0.69906	0.63816	0.5925
95% CI	0.01004843	0.0032431	0.0084961	0.00699565	0.00277728	0.00678783
Gapr2 Delivery Ratio	0.28584	0.56524	0.80744	0.87258	0.8856	0.89476
95% CI	0.00810256	0.00817832	0.01041319	0.00945395	0.0091256	0.00803473
Gapr Delivery Ratio	0.5581	0.92724	0.975	0.98388	0.98622	0.98692
95% CI	0.01305195	0.00717731	0.0033155	0.00233849	0.00154686	0.00173922
MaxProp Delivery Ratio	0.55646	0.92696	0.97532	0.98428	0.98606	0.98716
95% CI	0.0123005	0.00660047	0.00371816	0.00184461	0.00206579	0.00199363
Prophet Delivery Ratio	0.40228	0.72046	0.83946	0.78968	0.75418	0.72826
95% CI	0.01052394	0.00539131	0.00399326	0.00728858	0.00942659	0.00888575
Vector Delivery Ratio	0.27392	0.52616	0.7653	0.84244	0.86594	0.88158
95% CI	0.00865271	0.00994789	0.00843372	0.00538344	0.00822343	0.00897544

Graph11, Transmit Speed vs Overhead Ratio for 50MB buffer, slow message generator						
Transmit Speeds	0.125	0.25	0.5	1	1.5	2
Epidemic Overhead	14.62	24.5464	59.197	138.75282	220.539	304.6864
95% CI	0.3151585	0.25532294	1.19355442	3.28367601	2.80389989	6.66690343
Gapr2 Overhead	4.2805	5.10278	7.11044	9.9996	11.63884	12.92444
95% CI	0.14267381	0.16004344	0.15261583	0.20423701	0.1589332	0.1772321
Gapr Overhead	11.35058	18.07804	40.625	67.84864	80.49342	87.23576
95% CI	0.2648675	0.10702649	0.60504333	0.70397506	0.97921247	0.95177953
MaxProp Overhead	11.35056	18.0769	40.60632	67.85376	80.66184	87.47874
95% CI	0.27696454	0.14113539	0.52339151	0.80362893	0.86688691	0.95700815
Prophet Overhead	13.95624	20.56384	44.32314	93.96176	137.08376	176.25736
95% CI	0.45098329	0.47356029	0.96352434	1.80509302	1.65903652	3.26327794
Vector Overhead	4.33912	5.21078	7.06316	9.8009	11.1835	12.5434
95% CI	0.14379004	0.13686183	0.1561013	0.19740636	0.30800382	0.3043251

Graph 12, Transmit Speed vs Latency for 50MB buffer, slow message generator						
Transmit Speeds	0.125	0.25	0.5	1	1.5	2
Epidemic Latency	10098.52	8096.42	5149.02	3285.9	2871.36	2652.48
95% CI	253.626435	41.7064568	163.941676	76.0161566	27.9664798	101.547257
Gapr2 Latency	8458	7088.48	4890.32	3958.36	3731.36	3636.04
95% CI	429.360238	277.265648	172.572478	113.138608	114.320059	117.148633
Gapr Latency	8474.32	4558.7	1725.64	1111.64	964.84	898.54
95% CI	314.912629	132.782801	28.666389	17.5397071	25.9327404	25.1438646
MaxProp Latency	8630.8	4578.7	1728.24	1109.74	965.38	901.2
95% CI	308.067223	122.476418	21.7806058	18.2473076	24.6410715	24.6035964
Prophet Latency	9720.42	7416.82	4057.68	2594.44	2310.58	2160.4
95% CI	170.017774	261.03282	87.9095655	66.3904631	70.9619611	68.5497493
Vector Latency	8315.28	7058.04	5114.92	4188.66	3981.64	3833.98
95% CI	371.219364	274.154934	221.48497	145.100023	127.868594	115.482475

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C:
Random Mobility Simulations, 250m Duration, Aggregate Data

This appendix includes the means and confidence intervals of the performance measurements for the 250m duration random mobility simulations.

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.0642	0.0728	0.09772	0.07936	0.11806	0.1322	0.12424	0.15872	0.12642
95% CI	0.05129416	0.02319806	0.03828317	0.02882775	0.05199739	0.04093117	0.05401673	0.02519182	0.04117485
Overhead	4.90078	4.63432	4.50534	4.61868	3.985	4.08692	3.88346	3.66986	4.16038
95% CI	1.36741855	0.7184562	0.88804818	1.74856214	0.64332747	0.81505273	0.98998767	0.68871363	0.6034099
Latency	2158.68	2351.7	2244.66	2237.38	2459.88	2237.48	2367.74	2367.06	2498.14
95% CI	292.739181	324.43163	280.33979	453.403931	113.489921	104.55977	255.187014	390.965499	338.318499

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.05278	0.0828	0.10726	0.15896	0.15942	0.1759	0.17338	0.16928	0.21244
95% CI	0.01388165	0.0130907	0.02328239	0.03761144	0.02565344	0.03494846	0.03939746	0.00965577	0.03044661
Overhead	10.18622	10.6339	9.18714	10.56978	8.9052	9.9693	9.07792	9.47482	9.34406
95% CI	1.67436428	2.48565714	1.02646381	0.80469548	0.7630933	1.06360528	1.1449136	1.21871169	0.57172388
Latency	2616.82	2665.36	2707.3	2617.3	2772.38	2731.26	2704.94	2607.52	2740.12
95% CI	314.297897	223.934005	330.440305	172.591742	289.628229	206.214555	242.496777	212.717119	196.864727

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.05918	0.10062	0.13096	0.15706	0.19358	0.19952	0.1846	0.21642	0.228
95% CI	0.01123233	0.02508596	0.01319991	0.02278277	0.02333951	0.01813806	0.01600061	0.04561809	0.01175929
Overhead	14.72148	14.60636	14.37062	15.62758	15.91122	15.97056	14.8195	15.43414	15.00882
95% CI	2.88950143	1.0956141	0.72801962	1.79565146	0.75211715	2.1600923	1.16185255	1.75252596	1.58080475
Latency	2766.96	2685.82	2800.4	2914.38	2782.38	2989.38	2940.22	2819.22	3030.02
95% CI	119.401824	152.19078	216.214685	144.956205	141.441093	90.6561801	312.81947	139.220125	117.177862

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.0571	0.09702	0.15194	0.18508	0.21852	0.24016	0.2283	0.23748	0.26092
95% CI	0.00701293	0.02371447	0.00743241	0.01096479	0.01696595	0.02071349	0.01858469	0.02517406	0.01000827
Overhead	20.19684	20.55664	20.7916	21.00682	22.2726	22.83572	21.87244	22.24458	22.22468
95% CI	1.80261262	1.70996318	1.97731618	2.35493162	1.20871875	0.58580013	1.17021079	1.42594581	0.97582503
Latency	3043.02	2714.64	3030.56	3217.06	2989.02	3127.4	3053.72	3080.64	3081.32
95% CI	309.340797	371.976892	214.138434	264.738577	96.67623	180.68603	299.869731	90.8830397	183.756295

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.0772	0.13056	0.20438	0.2452	0.28684	0.29668	0.32382	0.32788	0.32608
95% CI	0.0107316	0.02084895	0.02155829	0.0162748	0.02591024	0.0131615	0.01065038	0.01344818	0.02702306
Overhead	27.47072	30.89092	34.29402	34.42246	34.45192	37.07776	36.58126	38.87606	36.41082
95% CI	2.61411098	0.72069358	1.76556556	1.66149712	2.47249445	4.45201806	2.28955857	1.12357434	2.60699321
Latency	3043.28	3212.66	3178.04	3283.94	3344.5	3133.88	3295.08	3248.1	3179.74
95% CI	472.713381	116.189118	143.899903	92.4252379	172.77575	92.8682797	150.821801	169.917751	75.4873138

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.08104	0.1556	0.24062	0.28254	0.32748	0.33748	0.36094	0.37796	0.3851
95% CI	0.00848325	0.01742104	0.02430104	0.01569251	0.00810547	0.01399822	0.02392704	0.0190137	0.0137584
Overhead	36.82964	43.21268	45.9231	49.219	51.82628	52.36796	55.7969	56.815	57.39426
95% CI	3.63038969	3.96649568	3.81955994	2.55575156	3.79444337	2.63880424	3.25071908	4.21282856	2.10799957
Latency	3000.74	3313.8	3455.3	3381.42	3346.26	3284.84	3305.44	3229.22	3297.26
95% CI	226.547585	175.569516	113.165423	162.570691	87.8311582	92.5533912	119.30329	171.341613	92.9168489

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09862	0.18812	0.27672	0.33484	0.3782	0.4003	0.42206	0.42566	0.4263
95% CI	0.01698538	0.02081577	0.01419638	0.00667438	0.02774779	0.01849654	0.00613668	0.02298556	0.00681129
Overhead	48.51354	56.28754	60.8226	67.2142	75.41614	74.26134	76.2395	79.15568	75.5935
95% CI	3.64084167	7.85915645	3.67011742	3.1242924	4.55302108	2.94537827	2.18230287	4.18449008	4.36440319
Latency	3064.66	3445.06	3558.06	3393.94	3408.5	3496.12	3375.86	3386.28	3289.72
95% CI	338.192665	118.134724	74.0154482	57.0533785	131.609168	140.247834	91.4927462	105.432977	46.1546

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.06764	0.07288	0.09734	0.07912	0.1168	0.13236	0.1258	0.16238	0.12706
95% CI	0.05634428	0.0240745	0.03632826	0.02678006	0.05159529	0.04065979	0.05853637	0.02803032	0.04464245
Overhead	2.70558	2.66502	2.4407	2.43748	2.44688	2.5306	2.331	2.17248	2.569
95% CI	0.39278397	0.33168254	0.33500094	0.13072429	0.15820599	0.15109072	0.23249924	0.18422028	0.37184453
Latency	2264.92	2408.68	2349.96	2199.2	2451.12	2362.34	2525.4	2401.24	2492.26
95% CI	372.226732	220.267951	331.017586	437.816832	131.344023	205.404473	256.032477	383.748472	347.75753

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.05348	0.0839	0.10868	0.16716	0.16568	0.17942	0.1809	0.17538	0.21924
95% CI	0.01465593	0.01445011	0.02420823	0.04320046	0.02717803	0.03453526	0.04334756	0.012353	0.03193029
Overhead	8.0713	7.08802	7.27524	6.90964	6.55672	7.1401	6.78084	6.69886	6.55904
95% CI	1.03463571	0.59614019	1.04079827	0.39952863	0.2748541	0.27914126	0.54496763	0.60222089	0.20902552
Latency	2635.16	2691.8	2742	2701.26	2867.12	2772.86	2825.68	2684.82	2814.3
95% CI	290.321665	193.058509	309.463593	168.69157	289.686096	204.150687	170.1058	206.221296	184.477393

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.05958	0.10188	0.134	0.16068	0.20086	0.20976	0.19294	0.22814	0.24174
95% CI	0.01152072	0.02597143	0.01502608	0.02458437	0.0253383	0.01952212	0.02013405	0.05280449	0.01413526
Overhead	12.30874	11.90702	11.54038	12.0214	11.72166	11.56206	11.24668	11.26996	10.66922
95% CI	0.8034892	1.01920797	0.69188804	1.29617355	0.35773229	0.3415811	0.3238096	0.39451864	0.55010842
Latency	2770.08	2707.76	2831.98	2923.98	2858.2	3068.76	3003.06	2895.06	3117.26
95% CI	122.050541	148.662136	249.531458	146.251668	145.607899	128.239178	275.661808	166.026451	158.812846

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.05726	0.09868	0.15532	0.1915	0.23446	0.2582	0.23952	0.25476	0.28174
95% CI	0.00723693	0.02490091	0.00945734	0.01330169	0.01911108	0.02082015	0.02050038	0.02647624	0.01391231
Overhead	18.16918	17.58356	16.62422	16.58388	16.4611	16.35258	16.22052	15.9993	15.78714
95% CI	0.91294462	0.68353256	0.88107245	0.89395453	0.62824428	0.30885538	0.72381791	0.42147423	0.42451279
Latency	3023.44	2744.88	3059.28	3259.64	3094.32	3182.74	3100.56	3112.86	3165.16
95% CI	273.583373	338.703867	209.991828	215.048858	118.296826	149.988809	324.86786	138.62202	168.510166

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.07752	0.13256	0.21296	0.26226	0.31044	0.3209	0.35666	0.36738	0.35854
95% CI	0.01067698	0.02259523	0.02178469	0.02036747	0.02860279	0.01855314	0.00905976	0.02068918	0.03760046
Overhead	25.92604	26.6569	25.98492	26.25332	25.44154	26.46186	26.02908	26.60256	25.26668
95% CI	3.14993621	1.78580215	1.19194544	1.42565976	1.13614616	1.53973813	1.26236185	0.55439539	1.08279219
Latency	3066.08	3276.9	3242.78	3363.8	3416	3177.28	3325.52	3256.06	3228.06
95% CI	461.50937	63.7992046	152.630708	79.5585973	177.516738	106.881797	155.393888	182.178617	51.3878873

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.08104	0.15866	0.25128	0.30592	0.35966	0.37484	0.40374	0.42924	0.44454
95% CI	0.00848325	0.01830039	0.02936029	0.02139995	0.01775025	0.0131099	0.02594188	0.02465003	0.01973232
Overhead	35.17756	36.6934	35.6263	36.34858	36.31658	36.40972	37.518	37.83774	37.95386
95% CI	2.86080628	2.16664778	2.36564072	1.86109696	0.54363372	1.38497473	1.71814064	2.53732949	0.84208844
Latency	3000.74	3313.6	3439.34	3385.96	3383.62	3322.86	3348.3	3255.28	3315.46
95% CI	226.547585	153.54116	119.683054	142.924288	78.2989182	84.6232266	115.80893	138.641455	65.0596675

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.099	0.19366	0.29284	0.3683	0.4277	0.44942	0.48766	0.49168	0.49496
95% CI	0.01720267	0.02236162	0.01798392	0.00956839	0.03829788	0.01558086	0.01035598	0.03249892	0.00345852
Overhead	46.03968	44.99448	47.39692	47.84972	50.46964	49.81922	50.15434	50.80056	50.06278
95% CI	3.8315937	2.907369	0.95187933	2.36477119	1.40779919	1.59957982	2.00020728	1.94660195	1.5837469
Latency	3067.94	3471.8	3573.44	3424.04	3363.56	3419.54	3294.02	3342.2	3245.6
95% CI	337.514375	117.268259	127.183701	39.5520641	94.3902681	137.76647	86.3016112	91.749207	84.6207269

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.06114	0.0721	0.09594	0.07848	0.11386	0.13024	0.12266	0.15542	0.1208
95% CI	0.04870525	0.0222792	0.03670873	0.02672566	0.04896205	0.03934148	0.05754679	0.02440966	0.03917217
Overhead	0.81772	0.9179	0.81122	1.33976	0.74634	0.83698	0.8096	0.58318	0.76582
95% CI	0.70830876	0.44541674	0.33782555	0.44290939	0.325815	0.33654593	0.19626538	0.20857969	0.18605566
Latency	2171.9	2349.02	2354.42	2212.72	2439.16	2313.3	2463.82	2437.9	2459.88
95% CI	351.072366	332.873008	336.173509	391.28367	105.235506	209.828446	162.061563	327.996557	349.412045

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.04826	0.07436	0.09626	0.13798	0.13486	0.14692	0.14808	0.144	0.17764
95% CI	0.01171215	0.00964817	0.0196476	0.02861117	0.01878398	0.02593793	0.03637532	0.00524682	0.02798994
Overhead	1.90564	2.07206	1.76952	1.98702	1.69238	1.94916	1.77566	1.914	1.54328
95% CI	0.46485066	0.60228134	0.33473795	0.65124071	0.25488737	0.27958881	0.32549908	0.35017208	0.31378189
Latency	2469.56	2436.98	2567.16	2477.7	2608.38	2497.74	2531.54	2458.24	2628.38
95% CI	342.395484	183.63438	205.218559	192.848367	149.782652	223.311489	73.370244	206.013822	150.078287

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.05108	0.08024	0.10524	0.11854	0.14708	0.14966	0.13688	0.16436	0.16716
95% CI	0.0061396	0.02036081	0.00924872	0.01527733	0.01506244	0.0098299	0.01167809	0.03453233	0.00954794
Overhead	2.90094	3.0773	2.81022	3.26632	3.10112	2.92036	3.10894	3.08472	2.57624
95% CI	0.57262944	0.38685836	0.43487289	0.65682116	0.39192456	0.32206274	0.3830259	0.293121	0.61289996
Latency	2518.7	2331.86	2514.18	2540.88	2546.9	2648.16	2614.1	2499.7	2703.26
95% CI	159.833364	222.951282	226.297825	224.400523	227.852929	104.677256	167.60457	64.0441576	129.479817

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.04576	0.0718	0.10492	0.1314	0.15064	0.16344	0.15384	0.16118	0.1703
95% CI	0.00440929	0.0130151	0.0074547	0.01058609	0.01227895	0.00951267	0.01547884	0.01425166	0.00596671
Overhead	3.85404	4.75638	4.07354	4.5536	4.23972	4.3555	4.1178	4.26432	3.8445
95% CI	0.55163681	0.47125013	0.53175074	0.44598137	0.57860236	0.54949846	0.29374404	0.36148835	0.31560732
Latency	2680.94	2306.88	2641.36	2730.16	2664.06	2771.82	2701.34	2646.74	2751.58
95% CI	274.153322	283.577121	251.293684	120.285615	189.010605	202.613016	304.542275	226.228136	180.654839

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.05506	0.082	0.11982	0.14744	0.16348	0.17308	0.18998	0.19656	0.18896
95% CI	0.00872585	0.01136323	0.01727036	0.00925039	0.01596944	0.01079902	0.011092	0.00783749	0.00898457
Overhead	5.79488	6.48892	7.20224	6.74092	6.95636	7.7364	6.91698	6.80166	6.62116
95% CI	1.09795415	0.69725935	1.0166749	0.49384392	0.51932533	0.51297324	0.37140633	0.50261576	0.46889055
Latency	2555.1	2765.06	2696.04	2832.76	2948	2668.28	2875.66	2784.52	2713.44
95% CI	487.754074	139.262714	218.764489	223.84691	157.635766	230.48609	222.913162	219.797135	175.874885

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.0532	0.08744	0.1279	0.1488	0.17178	0.1865	0.19164	0.20986	0.2018
95% CI	0.00687976	0.01225934	0.01425698	0.00813152	0.01591664	0.00782268	0.01008751	0.01175735	0.0115828
Overhead	8.11452	9.0656	9.4829	10.77366	10.51976	9.58564	9.83458	9.35236	9.28198
95% CI	0.46257845	1.22126362	0.60982016	1.1603018	0.19416479	0.61152025	0.57516093	0.67651735	0.79465351
Latency	2466.88	2683.74	2891.6	2984.02	2872.62	2867.44	2989.08	2919.58	3029.16
95% CI	135.30201	192.342284	163.808097	184.456906	284.866173	80.1630663	109.955735	284.486303	214.573688

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.0592	0.09604	0.1264	0.16328	0.19318	0.2032	0.21972	0.21752	0.21664
95% CI	0.00637495	0.0097862	0.00662728	0.0083874	0.02136967	0.01187202	0.00937712	0.02198487	0.00526184
Overhead	10.61316	11.96918	13.59604	13.01574	14.14358	12.95644	12.7909	12.82764	11.93484
95% CI	1.45432652	1.21445907	0.83841418	0.67324897	0.57693164	0.64003664	0.5530052	0.84383605	0.60141884
Latency	2465.6	2904.18	2900.68	2925.06	3031.6	3075.16	3066.52	3131.06	3003.42
95% CI	308.006602	124.546448	219.18224	231.731704	186.850061	230.070101	271.578019	124.428254	125.37281

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.06764	0.07318	0.09812	0.07834	0.11666	0.13252	0.12636	0.16154	0.12902
95% CI	0.05650145	0.02316685	0.03553445	0.0261454	0.05113187	0.0403043	0.05945601	0.02858165	0.04495259
Overhead	2.71812	2.59236	2.39346	2.43848	2.43404	2.52354	2.31216	2.18066	2.51068
95% CI	0.40141058	0.33197003	0.3459718	0.12329768	0.14492755	0.14973514	0.24991196	0.15417813	0.26729356
Latency	2232.26	2398.54	2386.38	2205.98	2446.68	2327.82	2529.86	2425.5	2515.38
95% CI	374.449513	275.076356	350.572983	396.962589	148.956348	229.221121	178.402162	409.992782	389.630272

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.0534	0.08412	0.10892	0.16706	0.1653	0.17802	0.1816	0.17608	0.21876
95% CI	0.01456086	0.01475532	0.0239109	0.04351572	0.0272163	0.03166875	0.04334094	0.01393751	0.03133096
Overhead	8.07506	7.06346	7.26814	6.9012	6.57548	7.14356	6.72934	6.63832	6.51672
95% CI	1.02453359	0.54558987	1.00283774	0.38303682	0.2851245	0.28632673	0.53797514	0.63181192	0.21538079
Latency	2627.34	2708.24	2741.74	2706.62	2870.04	2754.26	2862.84	2702.56	2827.52
95% CI	305.328066	180.166304	308.833329	171.65002	289.930767	241.805124	140.888602	233.674284	198.942975

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.05958	0.10188	0.1336	0.1613	0.20234	0.20952	0.19272	0.2284	0.24332
95% CI	0.01152072	0.02579847	0.01462172	0.02472252	0.02735649	0.01739915	0.01891367	0.05370222	0.01436673
Overhead	12.26372	11.88844	11.5461	11.92556	11.5729	11.5221	11.2376	11.21886	10.52724
95% CI	0.72980893	1.00585897	0.66380087	1.37301846	0.44841857	0.25739516	0.32415851	0.40147994	0.48091723
Latency	2770.08	2706.6	2839.94	2935.64	2872.32	3055.18	3019.68	2891.66	3130.26
95% CI	122.050541	148.399638	222.815456	123.051887	149.163436	123.162289	297.392358	174.565216	157.583346

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.05726	0.09892	0.15594	0.19218	0.23454	0.25788	0.24032	0.2547	0.28274
95% CI	0.00723693	0.02520222	0.00940936	0.01328857	0.02073848	0.02118883	0.02211869	0.02666509	0.01437171
Overhead	18.16792	17.52914	16.5096	16.45664	16.30166	16.2361	16.09288	15.88788	15.58442
95% CI	0.91558016	0.75386079	0.8826938	0.89379839	0.79487754	0.40948865	0.70803558	0.35668503	0.47021087
Latency	3023.44	2756.96	3066.16	3285.9	3092.28	3207.68	3111.54	3106.34	3173.54
95% CI	273.583373	353.86141	219.617961	209.416548	91.6225673	157.938829	328.701371	137.153749	190.298457

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.07752	0.13304	0.214	0.2628	0.31202	0.32392	0.36284	0.37082	0.36166
95% CI	0.01067698	0.0226688	0.02218614	0.01755117	0.02940468	0.0208711	0.01058189	0.02055506	0.03834025
Overhead	25.92084	26.52106	25.82058	25.9983	25.23382	26.02666	25.3613	26.249	24.82946
95% CI	3.15901245	1.79760665	1.19806871	1.17858045	1.20130849	1.33202424	1.1318273	0.45779801	1.02044573
Latency	3066.08	3264.8	3254.46	3368.58	3399.34	3204.54	3386.2	3271.9	3238.94
95% CI	461.50937	63.4886006	154.990797	97.2473082	170.555296	99.9304454	127.223004	172.883848	25.7720636

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.08104	0.15882	0.25198	0.30844	0.36344	0.37906	0.40976	0.43438	0.45056
95% CI	0.00848325	0.01822015	0.02957793	0.02079461	0.01969149	0.0137545	0.02314278	0.02191576	0.02205016
Overhead	35.17852	36.64676	35.36734	35.7966	35.80212	35.87792	36.79734	37.33938	37.46802
95% CI	2.86090941	2.07743696	2.29212464	1.77813548	0.55841104	1.29903058	1.37379123	2.26129022	1.14976362
Latency	3000.74	3317.36	3451.62	3388.36	3393.28	3347.26	3381.64	3285.24	3337.68
95% CI	226.547585	156.578224	120.45514	140.614676	86.2096676	108.028524	120.34784	127.559052	59.8257063

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09892	0.19376	0.29582	0.37308	0.4326	0.45782	0.4977	0.50318	0.50904
95% CI	0.01707501	0.02272776	0.01919364	0.00960936	0.03671649	0.0183725	0.00776571	0.03334818	0.00313981
Overhead	46.06606	44.89402	46.6183	47.17446	49.80946	49.0047	49.23264	50.02568	48.85506
95% CI	3.77681994	3.01172093	0.72511701	1.9612231	0.94774975	2.02704923	1.55919338	2.174257	1.47687749
Latency	3067.94	3470.46	3580.62	3431.88	3391.18	3411.2	3356.62	3369.68	3297.3
95% CI	337.514375	102.3945	134.452625	42.3916818	96.9369048	151.732072	80.0406084	98.9272724	65.958753

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.06122	0.07366	0.0963	0.07816	0.11518	0.132	0.12446	0.15528	0.12634
95% CI	0.04837313	0.02392008	0.03861196	0.02816849	0.04896701	0.03959479	0.05775746	0.02629613	0.03936538
Overhead	2.90154	2.51772	1.70306	1.88178	1.60088	1.7869	1.75798	1.36826	1.61574
95% CI	1.28683926	1.01974675	0.37198896	0.49110659	0.32222093	0.83734043	0.32047769	0.18298418	0.29018746
Latency	2026.38	2351.94	2226.2	2181.64	2423.48	2275.8	2390.32	2365.76	2484.32
95% CI	298.269758	332.027673	307.905724	412.615671	146.475022	155.544652	173.370755	393.665537	367.670597

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.05064	0.07936	0.1021	0.14856	0.1458	0.16138	0.16308	0.15926	0.19608
95% CI	0.01351364	0.01124391	0.02216988	0.03016468	0.01936598	0.025037	0.03936857	0.00833807	0.02911753
Overhead	6.73074	5.08828	4.26404	4.30822	3.96674	3.8686	3.89018	3.90366	3.6532
95% CI	0.99460615	1.01629101	0.6569285	0.57842425	0.37458928	0.28523658	0.48138462	0.56016783	0.17559259
Latency	2559.8	2575.8	2664.62	2555.84	2674.4	2601.36	2655.6	2551.44	2702.52
95% CI	280.710008	226.363854	312.730666	126.391164	223.403104	213.023656	144.063313	185.405817	192.83264

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.05496	0.09124	0.11872	0.13738	0.16966	0.176	0.16224	0.19376	0.20046
95% CI	0.00747337	0.02516122	0.01021895	0.01991273	0.02147606	0.01155295	0.01626752	0.04003034	0.01383296
Overhead	8.53106	7.0978	6.4235	6.28832	6.08148	6.13536	6.17634	5.79536	5.55096
95% CI	0.94491033	0.26256602	0.50171613	0.71107564	0.26937152	0.39798704	0.45425124	0.21934794	0.21823436
Latency	2634.7	2517.54	2672.96	2720.2	2675.58	2876.02	2814.14	2698.3	2926.14
95% CI	119.909338	174.356837	204.777049	180.024958	244.890736	117.591766	224.429696	168.969203	117.833053

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.0525	0.08624	0.1242	0.15282	0.18464	0.19998	0.19204	0.20112	0.21874
95% CI	0.00606969	0.02173963	0.00739807	0.01206049	0.0124886	0.01513881	0.01904092	0.01963171	0.00960493
Overhead	11.55142	10.34138	8.93132	8.66112	8.2917	8.31146	8.1341	8.00894	7.88988
95% CI	1.09561861	1.06010341	0.55122763	0.46627019	0.52410879	0.32626203	0.49162771	0.37242472	0.30643013
Latency	2824.76	2549.1	2800.94	2959.48	2821.18	3021.34	2912.56	2916.42	2925.16
95% CI	294.176219	351.107803	208.964955	188.733016	198.137291	210.117089	243.459119	214.060401	168.857915

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.06516	0.10386	0.1558	0.18936	0.21954	0.2291	0.2546	0.26376	0.25718
95% CI	0.00963666	0.01734416	0.01414516	0.01265538	0.01539515	0.01223477	0.01151445	0.00864205	0.02966859
Overhead	15.58652	14.00286	12.94892	12.22046	11.7968	12.0866	11.90686	11.92408	11.64424
95% CI	1.45733928	1.8956273	1.0839458	0.68187593	0.33160245	0.29377537	0.23455625	0.72327416	0.38365031
Latency	2774.36	2970.24	3002.6	3107.06	3185.58	2990.96	3108.86	3032.04	2979.12
95% CI	433.09005	91.921975	211.426351	134.898724	113.410341	128.960794	173.743619	188.308858	91.4691789

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.06316	0.11644	0.16976	0.20522	0.23862	0.25744	0.2758	0.29692	0.29966
95% CI	0.00729126	0.01334834	0.02169022	0.0118331	0.01553984	0.00713524	0.01988059	0.01727929	0.01566714
Overhead	20.394	18.26554	16.6103	16.6245	16.40894	15.82918	15.80614	15.79534	15.89146
95% CI	2.1866553	1.04182144	0.980127	0.41461501	0.58569133	0.44107193	0.94716708	1.1243081	0.73274503
Latency	2705.52	3078.78	3163.14	3240.58	3102.42	3160.26	3211.5	3118.16	3199.02
95% CI	209.041403	178.766146	147.933885	219.031773	196.853528	87.0821586	91.6979413	219.343094	52.051787

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.07572	0.13006	0.18732	0.2366	0.27952	0.29464	0.32686	0.33094	0.3282
95% CI	0.00976595	0.0122324	0.00832948	0.00546953	0.02537038	0.00819333	0.0067848	0.0322516	0.0103977
Overhead	24.09382	21.84464	20.51996	20.00756	20.1737	19.75816	19.34658	19.55126	19.03442
95% CI	2.65383755	1.72738915	0.51505368	0.97432663	0.45805557	0.53129245	0.3106268	0.49960546	0.87982504
Latency	2788.4	3143.74	3287.48	3181.96	3291.98	3284.96	3294.32	3342.04	3139.52
95% CI	257.758776	125.881109	58.9740155	140.421002	134.286599	150.30117	71.0646905	63.2647778	148.537216

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.06148	0.0713	0.09568	0.07756	0.1129	0.12894	0.12252	0.15448	0.12042
95% CI	0.04884059	0.02235657	0.03767196	0.02590715	0.04798847	0.03798295	0.05579146	0.02396848	0.03860672
Overhead	0.73142	0.8555	0.75394	1.23922	0.73162	0.81016	0.77804	0.56848	0.73116
95% CI	0.6020279	0.4029509	0.32177225	0.39259044	0.30057728	0.33044917	0.18715228	0.19930194	0.18256303
Latency	2189.34	2384.14	2319.56	2173.3	2422.58	2320.62	2474.02	2430.84	2467.42
95% CI	333.159554	376.690609	360.597453	420.43703	116.664423	181.05526	183.055292	350.251433	346.233409

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.04826	0.07428	0.0964	0.13798	0.13488	0.14678	0.148	0.144	0.17716
95% CI	0.01171215	0.00965609	0.01967808	0.02861117	0.01888283	0.02603096	0.03620689	0.00499143	0.02793613
Overhead	1.90304	2.06452	1.76566	1.9843	1.69132	1.9212	1.75942	1.91414	1.52674
95% CI	0.47075148	0.60527969	0.33133497	0.64740595	0.25601518	0.25330661	0.32113288	0.34185505	0.31020793
Latency	2469.56	2436.98	2569.96	2477.7	2612.84	2494.56	2530.26	2462.02	2626.52
95% CI	342.395484	183.63438	204.699171	192.848367	180.033103	215.581429	73.3971357	208.349597	151.758171

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.05108	0.08024	0.10524	0.1187	0.14686	0.14942	0.1369	0.16436	0.16668
95% CI	0.0061396	0.02036081	0.00924872	0.01500822	0.01482915	0.00995731	0.01157668	0.03435329	0.0095933
Overhead	2.90094	3.0773	2.80868	3.2681	3.08628	2.9149	3.11236	3.06818	2.59666
95% CI	0.57262944	0.38685836	0.43518141	0.65853618	0.35315203	0.32140776	0.35774564	0.31992462	0.61516653
Latency	2518.7	2331.86	2514.18	2532.16	2539.56	2641.52	2600.32	2499.7	2702.38
95% CI	159.833364	222.951282	226.297825	211.204945	218.341544	107.396497	188.624274	64.0441576	130.084614

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.04576	0.0718	0.10486	0.13132	0.1511	0.16314	0.1536	0.16078	0.17016
95% CI	0.00440929	0.0130151	0.00738664	0.01061239	0.01240123	0.01009148	0.01523436	0.01424993	0.00552985
Overhead	3.85246	4.75898	4.07508	4.55054	4.21196	4.36512	4.12564	4.27608	3.84654
95% CI	0.55208395	0.47718442	0.53236776	0.44720535	0.57278261	0.55405359	0.30354485	0.39248198	0.26504063
Latency	2680.94	2306.88	2641.36	2730.16	2666.18	2759.96	2697.5	2648.04	2751.78
95% CI	274.153322	283.577121	251.293684	120.285615	184.513015	204.743076	300.144446	231.204566	164.076159

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.05506	0.08184	0.11966	0.14688	0.16362	0.17286	0.1896	0.19624	0.18834
95% CI	0.00872585	0.01116217	0.01716492	0.00878846	0.01622231	0.01018818	0.01104983	0.00790955	0.01001049
Overhead	5.78592	6.49416	7.19488	6.77038	6.91922	7.71764	6.9372	6.8245	6.64548
95% CI	1.10861618	0.67737072	1.0045269	0.52488204	0.4799506	0.4723352	0.38010765	0.53261313	0.4680599
Latency	2555.1	2764.08	2687.44	2821.36	2957.18	2652.34	2880.18	2774.58	2690.08
95% CI	487.754074	139.281373	212.195351	233.297565	152.253899	208.907945	231.233008	237.61536	156.590586

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.0532	0.0876	0.12804	0.14882	0.17162	0.1868	0.19056	0.20876	0.19994
95% CI	0.00687976	0.01248129	0.01452961	0.00874237	0.01481767	0.00830004	0.009282	0.01259823	0.01010278
Overhead	8.1233	9.06502	9.45746	10.74274	10.474	9.47502	9.86038	9.3009	9.22824
95% CI	0.46318933	1.21573334	0.60358853	1.22219171	0.25701205	0.65093463	0.52750379	0.61422421	0.82376168
Latency	2466.88	2666.66	2895.32	2976.02	2870.58	2862.2	2961.06	2944.68	3026.12
95% CI	135.30201	190.844178	164.58106	186.822903	285.88924	78.4600004	126.682289	305.509325	209.367812

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.05912	0.09596	0.1264	0.16314	0.1933	0.20172	0.21868	0.21666	0.2144
95% CI	0.00647782	0.00990023	0.00649569	0.00793756	0.02163562	0.01056258	0.0087709	0.02089032	0.00678635
Overhead	10.6431	11.98242	13.60442	12.91934	14.07954	12.9627	12.703	12.7966	11.9013
95% CI	1.45325598	1.21050722	0.8650486	0.66530543	0.47773509	0.66329153	0.52236857	0.84095472	0.59274557
Latency	2453.38	2905.5	2900.68	2938.22	3038.14	3074.36	3063.64	3102.02	2963.86
95% CI	328.566268	123.490585	219.18224	215.079895	195.127165	208.570705	255.254647	151.962354	110.386167

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D:
Random Mobility Simulations, 500m Duration, Aggregate Data

This appendix includes the means and confidence intervals of the performance measurements for the 500m duration random mobility simulations.

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.08962	0.12386	0.19908	0.17622	0.26848	0.28312	0.29016	0.2808	0.28022
95% CI	0.03103757	0.03396786	0.05850931	0.02897187	0.07266645	0.03886082	0.06988164	0.03500365	0.06481659
Overhead	4.32462	3.96382	3.26266	2.6973	2.26328	2.38626	2.56022	2.16324	2.44214
95% CI	1.62392772	0.44345984	0.64256207	0.60318409	0.2711523	0.16461244	0.43379968	0.24785546	0.26115665
Latency	2473.3	2495.82	2451.92	2503.78	2523.74	2410.9	2477.88	2346.92	2507.86
95% CI	285.463247	196.469549	229.392622	237.959499	149.172886	155.42251	191.239933	124.534985	257.102794

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09484	0.13072	0.22962	0.31986	0.36144	0.40008	0.4003	0.43324	0.46232
95% CI	0.01183758	0.01994167	0.035812	0.04171434	0.02776119	0.02866608	0.02493724	0.03334312	0.03884566
Overhead	9.6072	9.73132	9.07692	8.63602	7.87068	7.7574	7.09422	6.73232	6.78762
95% CI	0.79067164	0.86552879	0.35959368	0.6701357	0.61702463	0.65833862	0.47041703	0.62260933	0.69621611
Latency	2737.68	2795.46	2910.94	2713.48	2776.34	2821.86	2763.38	2782.48	2700.64
95% CI	284.661478	207.715485	102.052623	238.747784	89.6520135	120.121134	100.027774	116.965049	99.0336945

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09486	0.17364	0.28068	0.3538	0.43426	0.47392	0.48132	0.52376	0.53034
95% CI	0.01751752	0.01150416	0.01727268	0.03756841	0.02054909	0.03301999	0.02065353	0.01839903	0.02603478
Overhead	15.32424	16.59834	15.98278	15.47798	14.45774	14.71814	13.15284	13.88048	13.55662
95% CI	3.22429587	2.52779797	1.19471056	0.61553623	1.10805266	0.99282691	0.96735259	0.7401014	1.04267948
Latency	2996.96	3028.74	3079.52	2990.2	2982.62	2821.74	2879.4	2779.26	2727.18
95% CI	205.289692	145.142876	135.387558	120.785867	77.9981422	78.834404	114.607995	127.285146	109.380278

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09484	0.21584	0.33586	0.43476	0.50104	0.53718	0.55586	0.58744	0.59134
95% CI	0.01740893	0.01513661	0.00904307	0.01478542	0.00952433	0.01338682	0.01837589	0.01631484	0.01190641
Overhead	20.45534	21.62782	22.12558	22.86872	23.04026	22.65368	23.34938	22.29052	21.5692
95% CI	1.30752653	1.40791684	1.95751132	1.0156944	0.97193281	1.29659536	0.69213385	0.79330142	0.37845575
Latency	3152.96	3284.24	3168.66	3076.62	2951.78	2901.36	2751.4	2734.38	2744.38
95% CI	137.143542	110.70142	117.197023	98.09238	131.336933	92.6488241	44.1535438	64.1884518	48.165771

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.13132	0.254	0.41368	0.53194	0.57944	0.62182	0.64292	0.65796	0.66274
95% CI	0.01484428	0.01005763	0.01278116	0.01627832	0.01947452	0.01175047	0.00842774	0.0138178	0.0112247
Overhead	31.97156	34.75996	39.94694	43.43842	45.3116	45.77788	46.0928	44.52818	44.4925
95% CI	2.9216785	1.34087786	1.32135013	1.31164947	1.68302307	1.58011756	2.11748329	1.57512475	2.43631457
Latency	3476.16	3451.02	3361.94	3062.96	2929.08	2770.8	2697.98	2690	2643.02
95% CI	256.680878	37.1112079	108.853889	111.018111	56.7548519	58.502538	73.2361762	117.094037	99.7543428

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.1663	0.31028	0.49362	0.5763	0.62086	0.65074	0.67746	0.68006	0.69868
95% CI	0.01284147	0.01676612	0.01971837	0.00924852	0.01045319	0.00774488	0.00918482	0.00742328	0.01588348
Overhead	39.93064	49.97466	63.59174	69.57272	76.10604	78.28472	78.28976	80.3503	76.52848
95% CI	4.83533462	2.70782653	3.51874395	2.31121592	1.04461257	3.00325228	2.59908736	1.42534375	2.06134726
Latency	3544.38	3524.48	3273.54	2987.56	2845.62	2709.24	2655.48	2573.1	2580.56
95% CI	109.645059	100.824994	110.917911	125.683009	66.9341157	76.6266497	44.8383008	71.2530808	62.9779324

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.18482	0.37732	0.53154	0.60124	0.65008	0.66764	0.68284	0.69322	0.6991
95% CI	0.00785195	0.01675509	0.01811615	0.0103875	0.00699197	0.01660873	0.00661265	0.0107896	0.00628385
Overhead	51.625	65.7737	89.07236	104.84106	113.14036	123.60138	123.29096	122.3634	120.5677
95% CI	2.91799018	4.97183949	1.27467414	0.89032872	1.95145783	2.88549686	3.45099143	2.85187828	2.17163077
Latency	3495.92	3584.04	3287.66	2976.1	2755.14	2551.86	2506.28	2445.94	2458.3
95% CI	92.6913078	103.875583	44.3108531	74.8328137	116.708559	52.1712638	38.4058858	108.456677	88.9434005

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09164	0.1216	0.20288	0.17842	0.27398	0.2954	0.29922	0.2911	0.28748
95% CI	0.03128467	0.03211424	0.05931004	0.03251846	0.08117469	0.03814896	0.07997278	0.04405364	0.06533283
Overhead	2.51986	2.74102	2.32188	1.93254	1.83562	1.68018	1.73238	1.65534	1.83646
95% CI	0.26050203	0.29267966	0.41463532	0.19690481	0.2157394	0.23808575	0.27412496	0.18847482	0.03335994
Latency	2666.18	2527.46	2640.06	2580.82	2647.28	2566.62	2595.46	2579.24	2584.9
95% CI	211.782507	207.478043	194.414237	252.313963	173.554185	110.817087	194.759336	40.362339	257.933729

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09656	0.13854	0.23854	0.34692	0.39468	0.43932	0.43854	0.47006	0.51148
95% CI	0.0112115	0.02406351	0.03777211	0.05220977	0.02774994	0.04352073	0.03554028	0.04538961	0.03474028
Overhead	7.19114	7.19754	6.88394	6.12478	5.55318	5.45622	5.0242	4.75568	4.6348
95% CI	0.62847781	0.53895592	0.3230126	0.25896574	0.22686649	0.18153128	0.11575487	0.26995168	0.4237046
Latency	2796.3	2900.92	2970.06	2859.22	2899.16	2959.02	2847.96	2947.2	2895.76
95% CI	312.582681	229.549556	67.2294543	225.96682	95.8485021	130.255688	119.803953	101.494226	97.9880006

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09734	0.18116	0.3033	0.40304	0.50508	0.55536	0.55878	0.62164	0.6304
95% CI	0.01982134	0.01237824	0.02475169	0.046789	0.0316212	0.05433886	0.03145195	0.03359022	0.03316243
Overhead	12.39786	12.32166	11.79914	10.86126	9.96686	9.71764	8.70186	8.90524	8.43408
95% CI	1.18994453	0.90841691	0.97454459	0.34664176	0.30306655	0.44365527	0.52996226	0.35399823	0.44418952
Latency	3044.48	3104.82	3167.5	3138.9	3149.5	2969.8	3005.48	2954.34	2859.04
95% CI	160.798496	115.071278	116.033697	84.3229736	40.9956011	88.7306681	75.5920881	86.341054	91.9402548

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.0969	0.2282	0.36952	0.50628	0.61086	0.66226	0.69914	0.73152	0.73964
95% CI	0.01737602	0.01950986	0.01805217	0.03107589	0.02005917	0.00403462	0.02398625	0.02206956	0.01525673
Overhead	17.31284	16.6165	16.0845	15.68868	14.9874	14.16784	13.90908	13.30166	12.81646
95% CI	0.80042797	0.64145733	0.99745254	0.30564359	0.34213022	0.53767971	0.50855658	0.17352635	0.21871536
Latency	3176.94	3381.5	3282.42	3155.96	3054.22	3021.58	2861.94	2799.76	2781.06
95% CI	112.235893	143.694916	41.1074728	76.5135319	86.4683754	69.5989344	90.0823799	81.5697307	87.9103232

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.13398	0.27392	0.483	0.66502	0.75252	0.81914	0.85482	0.86852	0.88156
95% CI	0.01464751	0.01169576	0.01663274	0.02479083	0.04221653	0.02226738	0.00705999	0.02595884	0.01462902
Overhead	27.14044	26.64558	28.03154	27.85508	27.06338	25.72286	24.7197	23.77298	22.71736
95% CI	1.73516756	0.93705483	0.39829719	0.67205626	0.54814391	0.33949702	0.31919506	0.25418549	0.59215352
Latency	3513.8	3497.22	3391.38	3120.76	2908.22	2751.58	2604.94	2552.72	2499.56
95% CI	279.369422	60.6652379	105.769817	72.7720022	99.3738505	102.944456	59.6325911	127.737704	130.996519

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.16952	0.34254	0.60164	0.75258	0.84956	0.89068	0.92192	0.93878	0.94872
95% CI	0.01402023	0.01669003	0.03269739	0.01225653	0.01780524	0.00300821	0.01155973	0.0049997	0.0091464
Overhead	34.4897	37.64288	41.77078	42.02112	41.6224	39.18554	37.42278	35.68788	33.6567
95% CI	2.12543515	1.54741273	1.49401897	0.98979792	0.50689642	0.24693605	1.09256885	1.14298843	0.94355085
Latency	3557.74	3553.5	3226.44	2900.8	2710.92	2427.04	2307.96	2198.48	2130.64
95% CI	128.768629	114.937396	113.679519	68.5097942	32.5349519	82.5994089	58.7607356	69.95784	47.5782962

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.18812	0.42058	0.65994	0.81242	0.90574	0.94166	0.95776	0.96214	0.9703
95% CI	0.00985647	0.01432805	0.01643254	0.02456367	0.01571509	0.01368618	0.00886225	0.00560297	0.00392413
Overhead	44.84002	49.24796	57.24038	59.92662	57.54416	53.73776	50.15716	46.73004	44.08146
95% CI	1.64460302	2.24599349	1.17312547	1.32987209	0.83993442	1.74362852	2.44152301	2.35959147	0.87319889
Latency	3530.38	3556.58	3167.76	2757.94	2400.26	2104.74	1976.66	1883.16	1839.28
95% CI	79.6119634	118.138347	68.9352618	57.1737722	98.165161	78.7759082	103.620847	112.718381	54.1259781

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.0863	0.119	0.19328	0.17248	0.25308	0.27318	0.27818	0.27138	0.2685
95% CI	0.02842418	0.03173495	0.05894457	0.03078149	0.06691841	0.03938768	0.06623732	0.03275701	0.05748741
Overhead	0.80706	0.64568	0.58248	0.61012	0.35874	0.41402	0.45772	0.40808	0.49048
95% CI	0.63790537	0.17191578	0.12049537	0.09585496	0.09160729	0.09015669	0.11715428	0.18866623	0.13884225
Latency	2546.96	2534.84	2603.1	2524.08	2538.56	2464.42	2546.4	2442.26	2538.1
95% CI	273.740666	233.267481	202.919719	241.663688	170.237516	174.437493	135.588753	57.1346585	232.785052

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.081	0.10968	0.18122	0.26088	0.29164	0.32534	0.32706	0.35278	0.37976
95% CI	0.0090063	0.01658361	0.02020114	0.03617915	0.02155903	0.03492734	0.01540416	0.0309542	0.02964709
Overhead	1.50644	1.43836	1.48816	1.40038	1.36474	1.41902	1.2679	1.15514	1.09316
95% CI	0.48744065	0.41710516	0.17315674	0.14732159	0.26848116	0.14721661	0.13454262	0.11217076	0.15825682
Latency	2576.98	2557.74	2733.98	2582.32	2634.36	2681.22	2679.56	2633.84	2611.26
95% CI	332.926249	224.551748	144.113889	254.728746	32.0143842	141.880037	140.939433	102.102465	139.28109

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.0749	0.1265	0.19756	0.2537	0.3172	0.35614	0.35856	0.38912	0.40414
95% CI	0.01374696	0.00565687	0.01782376	0.03145319	0.01987702	0.04188423	0.01419121	0.020668	0.01565769
Overhead	2.38196	2.5564	2.6431	2.5016	2.30596	2.22598	2.12716	2.08588	1.90422
95% CI	0.52517429	0.43688909	0.23246936	0.20396795	0.11643283	0.15998464	0.2846011	0.13821391	0.12884585
Latency	2790.12	2694.14	2832.9	2752.3	2890.22	2693.16	2750.22	2723.46	2664.5
95% CI	337.553865	104.713099	163.293517	82.4650376	110.344924	113.560843	98.7592225	49.7141922	139.888513

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.06766	0.13264	0.2104	0.2807	0.34182	0.38088	0.39964	0.42726	0.41846
95% CI	0.00714258	0.01245027	0.00509173	0.01867076	0.00771928	0.0039213	0.0105495	0.01411671	0.01413679
Overhead	3.68026	3.43806	3.22798	3.56666	3.50462	3.32514	3.23804	3.09126	2.71102
95% CI	0.43193729	0.59826224	0.41223726	0.21072965	0.28993325	0.26780726	0.21777168	0.19609517	0.13311723
Latency	2739.72	2707.58	2787.8	2877.9	2924.14	2916.14	2775.98	2758.6	2685.72
95% CI	149.704075	191.544699	116.155553	74.863093	138.532794	69.8661989	161.923162	121.104373	159.51824

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.074	0.13258	0.22636	0.32422	0.39162	0.44326	0.46946	0.4789	0.4872
95% CI	0.00707945	0.00581088	0.01683041	0.01420854	0.0299802	0.01423287	0.0166758	0.02506046	0.02075875
Overhead	5.18832	5.10628	5.5211	5.52198	5.69908	5.51958	5.11826	5.07752	4.8205
95% CI	0.77148869	0.39867962	0.27506761	0.36328735	0.21440667	0.21810145	0.14234779	0.22088883	0.23960242
Latency	2751.7	2869.68	2947.66	3009.22	3012.74	2961.88	2929.32	2880.42	2917.62
95% CI	452.707547	74.1586939	270.549241	166.802692	107.637663	101.70762	69.1597886	68.8552296	67.6780646

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.08182	0.14504	0.25734	0.35538	0.4357	0.47832	0.51966	0.5417	0.54928
95% CI	0.00776865	0.00738914	0.01575937	0.0133305	0.01100481	0.00743897	0.00622598	0.01581882	0.01391692
Overhead	6.51396	7.26578	7.98422	8.29076	8.42926	7.85864	7.59256	7.39624	6.79952
95% CI	1.01259108	0.77138759	0.43885476	0.48152778	0.23789555	0.15047067	0.17678241	0.35608613	0.31102722
Latency	2783.38	2924.08	3027.46	3022.54	3143.04	2959.24	2985.7	2993.94	2929.1
95% CI	203.161038	140.425938	154.974025	138.371219	87.2663012	95.3899136	70.6588206	94.0258161	130.623619

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.08386	0.15798	0.28214	0.39608	0.47438	0.54336	0.57286	0.58702	0.5937
95% CI	0.00448555	0.0061931	0.00972488	0.0161224	0.02103328	0.02214797	0.0144837	0.01991706	0.01255445
Overhead	8.19784	9.14126	10.86804	11.19258	10.53972	10.4732	9.93948	9.8744	9.19164
95% CI	1.16281197	0.37037918	0.57544636	0.38647058	0.2792368	0.09736942	0.30939962	0.39632313	0.40377785
Latency	2754.94	3068.72	3210.96	3259.02	3123.04	3035.84	2975.5	2988.2	2954.3
95% CI	299.271529	196.880421	157.570412	67.4275639	100.219826	109.92228	141.592407	122.887405	113.315647

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09172	0.12282	0.20304	0.17834	0.275	0.29622	0.3007	0.2915	0.28718
95% CI	0.03117614	0.03195623	0.06135165	0.03205192	0.08186079	0.04242818	0.07908458	0.04422339	0.06478685
Overhead	2.4979	2.65706	2.30654	1.92542	1.82032	1.67864	1.7155	1.6503	1.83348
95% CI	0.22504003	0.28543964	0.39768188	0.19076667	0.23683062	0.25343607	0.26750731	0.20029996	0.03910555
Latency	2639.96	2605.22	2630.76	2553.52	2662.84	2559.22	2604.52	2564.3	2611.52
95% CI	213.844644	228.502524	186.504589	266.769045	153.715969	108.558291	196.975135	46.7450464	289.800902

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.0974	0.13878	0.24154	0.34596	0.39444	0.43798	0.4395	0.47208	0.51064
95% CI	0.01244046	0.02351089	0.04034813	0.05178138	0.02815755	0.04125027	0.03414906	0.04474364	0.03522919
Overhead	7.07788	7.10456	6.77286	6.09364	5.53654	5.46496	4.99128	4.7335	4.6088
95% CI	0.44811576	0.47276318	0.34543833	0.26118013	0.23199561	0.18157619	0.1008722	0.28169795	0.38900194
Latency	2786.74	2891.44	3019.56	2868.66	2913.68	2963.88	2845.14	2938.52	2896.46
95% CI	328.822988	229.894599	78.9784824	256.262082	67.785457	142.783054	103.127341	109.73136	119.490317

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.0978	0.18124	0.30446	0.40414	0.5082	0.55546	0.5598	0.62486	0.63612
95% CI	0.02046344	0.01175551	0.02185591	0.04981865	0.03412539	0.05644805	0.03235854	0.03417826	0.03126454
Overhead	12.35102	12.20366	11.6403	10.77588	9.83648	9.69808	8.6486	8.82358	8.2963
95% CI	1.12239076	0.77929532	0.86864374	0.43777408	0.2765204	0.50278781	0.49801593	0.32069223	0.41985757
Latency	3047.74	3102.92	3162.7	3141.2	3151.58	2969.36	3032.66	2976.16	2881.02
95% CI	154.873714	101.830238	145.38947	89.7513354	36.105864	98.7992472	105.44737	67.690875	90.2996726

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09696	0.22914	0.36994	0.5064	0.61178	0.6629	0.70086	0.7347	0.74374
95% CI	0.01729342	0.01965591	0.01688108	0.02518626	0.01476158	0.00227736	0.02527225	0.02224056	0.01751364
Overhead	17.2825	16.40598	15.91768	15.5888	14.87648	14.1142	13.81952	13.20276	12.63592
95% CI	0.77273152	0.54369622	1.01674208	0.27494251	0.26135852	0.53757336	0.50990392	0.1466478	0.27650504
Latency	3176.94	3385.54	3303.54	3174.9	3069.84	3036.46	2865.86	2813.26	2814.44
95% CI	112.235893	128.876746	38.6346781	67.6464654	90.7032483	95.9950216	90.1861035	98.5245666	91.8355242

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.13464	0.27478	0.48238	0.67206	0.7616	0.8272	0.86438	0.87266	0.88776
95% CI	0.01429189	0.0115175	0.01465929	0.02631257	0.04219612	0.02268789	0.00860408	0.02513229	0.0115122
Overhead	26.90526	26.2541	27.82754	27.51514	26.85158	25.57386	24.47012	23.62834	22.63956
95% CI	1.58211345	0.92760393	0.47514262	0.58275342	0.52581641	0.43441489	0.44718332	0.13439831	0.4805425
Latency	3516.84	3511.44	3413.7	3139.52	2931.16	2771.8	2627.36	2556.4	2504.96
95% CI	274.498023	59.4130501	89.0985763	71.7487309	105.371285	103.900882	52.4841605	132.392199	128.12388

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.16992	0.34554	0.60962	0.76396	0.8637	0.8999	0.92846	0.94366	0.95364
95% CI	0.01391072	0.01623698	0.03299486	0.01510521	0.02149406	0.00867211	0.01053721	0.00457741	0.00609636
Overhead	34.41604	37.09948	41.41826	41.99332	41.36978	39.22428	37.45902	35.76844	33.49876
95% CI	1.94028423	1.43507093	1.4676579	1.18068489	0.49707673	0.72420057	1.18328134	0.99133544	0.81409971
Latency	3563	3571.08	3273.14	2932.6	2704.44	2438	2299.82	2200.24	2125.14
95% CI	134.596309	117.871848	112.251551	103.800526	63.8211237	70.4618227	65.3921328	59.8131909	50.7005697

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.18906	0.4244	0.67972	0.82778	0.91826	0.949	0.96184	0.9659	0.97106
95% CI	0.01000063	0.01519971	0.0205423	0.02395188	0.01250365	0.01130283	0.00649028	0.00414872	0.00395044
Overhead	44.53066	48.66412	56.21428	59.4847	57.29336	53.70258	49.97424	46.6765	43.9319
95% CI	1.486033	2.18736054	1.03081546	1.14634996	0.91335611	1.73068264	2.13637882	2.45831556	0.80051042
Latency	3533.66	3582.46	3220.14	2753.62	2403.94	2095.3	1969.74	1885.8	1835.18
95% CI	81.7831812	109.50014	65.971729	61.4154377	90.8402807	64.6980581	92.0122654	108.09328	57.3767224

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.08812	0.12088	0.19608	0.17506	0.2609	0.2749	0.28254	0.27966	0.27334
95% CI	0.02933139	0.0335835	0.05249966	0.02698837	0.07066122	0.03640494	0.06298063	0.03686422	0.06302394
Overhead	2.12108	1.663	1.49902	1.25736	1.14298	1.11862	1.10998	1.02356	1.11736
95% CI	0.36663337	0.26393084	0.32574135	0.10065025	0.08412749	0.15647202	0.16282459	0.14762274	0.03573755
Latency	2456.1	2449.42	2451.06	2510.56	2538.06	2344.9	2489.62	2378.92	2470.12
95% CI	285.043404	178.430416	189.523821	226.151277	197.15575	96.9017368	157.674558	77.7663748	273.98196

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.08772	0.12612	0.20694	0.30072	0.34258	0.37968	0.38522	0.40562	0.44032
95% CI	0.00863591	0.02060944	0.0309321	0.03574806	0.02352531	0.03370576	0.03263856	0.03309666	0.0265908
Overhead	4.1423	4.1865	3.76578	3.50304	3.43246	3.26786	3.06336	2.933	2.82194
95% CI	0.54270828	0.27031507	0.12479674	0.12892909	0.14705028	0.156679	0.1524498	0.10071103	0.25139565
Latency	2693.32	2722.94	2872.74	2692.38	2795.56	2774	2763.92	2729.92	2739.38
95% CI	312.63943	230.855981	89.6291732	257.707456	41.7752017	195.715127	88.7792155	108.326211	123.503204

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.08532	0.1501	0.2441	0.31484	0.4064	0.45388	0.44762	0.50304	0.51368
95% CI	0.01433795	0.00550997	0.02230438	0.03639551	0.02616313	0.03776557	0.0297454	0.01236677	0.02153181
Overhead	7.67374	6.47362	5.87366	5.74004	5.57808	5.44222	5.1533	5.11742	4.87998
95% CI	1.46671699	0.69969917	0.44069767	0.23116992	0.20544237	0.14180741	0.30202867	0.15749949	0.18975896
Latency	2905.26	2883.14	3034.54	2922.78	3011.94	2851.76	2877.46	2866.84	2842.64
95% CI	247.122488	125.356926	131.62024	90.1553205	99.5431302	66.1904793	99.1212709	65.2772228	140.236626

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.08064	0.1715	0.2808	0.37822	0.47172	0.5142	0.55006	0.57328	0.58538
95% CI	0.01169634	0.0116216	0.01619827	0.02108657	0.01004855	0.01253257	0.01120683	0.01616462	0.01795695
Overhead	9.91162	8.05294	7.54662	7.597	7.65568	7.53448	7.54912	7.35402	6.99152
95% CI	0.86324837	0.42980026	0.48492623	0.15349603	0.25943218	0.21514238	0.15477321	0.13843119	0.16827025
Latency	2966.06	3048.5	3101.48	3032.22	3048.1	3021.44	2882.16	2839.96	2844.84
95% CI	153.382379	148.657229	151.646176	113.509617	60.8727975	95.3071262	104.141864	75.8128192	55.326887

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09834	0.1835	0.33382	0.48808	0.56448	0.62066	0.65938	0.67854	0.68886
95% CI	0.00775681	0.01247857	0.01758775	0.01700788	0.03903847	0.01838277	0.0106917	0.02292511	0.01912769
Overhead	13.81314	12.22336	11.72864	11.48796	12.0194	12.18342	12.4052	12.09244	11.674
95% CI	0.70083344	0.74842128	0.37165987	0.30630042	0.20594113	0.34622376	0.32317017	0.25770737	0.55855148
Latency	3129.1	3147.68	3312.62	3178.82	3016.18	2873.64	2772.82	2782.48	2782.28
95% CI	342.563508	107.781081	118.827164	103.361511	79.5084453	98.5877024	60.1173856	61.0035171	121.111359

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.11498	0.2183	0.4243	0.55022	0.6454	0.69102	0.72928	0.74256	0.75742
95% CI	0.00965386	0.01140953	0.03121534	0.0188678	0.01275764	0.01787589	0.00757289	0.0100618	0.00524588
Overhead	16.82832	15.73916	15.28044	15.77466	16.7079	17.15696	17.86532	18.03556	17.34
95% CI	1.60746683	0.5345524	0.33587511	0.40920948	0.35139223	0.38568569	0.80830829	0.25869236	0.13052663
Latency	3107.74	3305.62	3301.58	3115.68	2982.24	2767.34	2708.24	2655.46	2626.18
95% CI	103.849903	132.576307	50.6412262	85.4597344	87.2236074	66.7747735	95.088526	73.0754698	74.4671107

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.12364	0.2585	0.46122	0.60476	0.70136	0.74626	0.77802	0.77764	0.79044
95% CI	0.00598611	0.01263548	0.01282114	0.01561095	0.01805289	0.01638801	0.008927	0.01264099	0.00554322
Overhead	20.63148	19.4072	19.29628	20.7153	21.93166	23.61256	23.90504	24.54002	24.10592
95% CI	0.8544354	1.32439291	0.32979902	0.53099774	0.37791961	0.49173423	0.41281357	0.37919233	1.04992984
Latency	3200.76	3437.8	3356.66	3116.04	2814.1	2603.88	2543.86	2461.54	2430.56
95% CI	145.331445	183.583404	92.7463199	72.6229137	92.9664607	48.4656563	64.2917072	81.3533324	72.2815519

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.08582	0.11938	0.19306	0.17108	0.25246	0.2713	0.27826	0.26958	0.2667
95% CI	0.02841967	0.03065934	0.0586172	0.03143342	0.06745034	0.03791327	0.06612117	0.03248734	0.05726874
Overhead	0.77624	0.60002	0.56296	0.59924	0.35606	0.40684	0.44662	0.38634	0.4725
95% CI	0.58097448	0.20298802	0.1171851	0.10756396	0.09316759	0.08852176	0.11436135	0.17226087	0.14110004
Latency	2546.96	2556.08	2599.18	2520.34	2536.14	2454.46	2560.3	2450.14	2530.76
95% CI	273.740666	258.237903	196.094331	228.12839	172.856337	174.535574	157.008911	56.1773364	225.778394

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.081	0.10976	0.18076	0.26112	0.29126	0.3252	0.3266	0.3534	0.37968
95% CI	0.0090063	0.01646928	0.02024767	0.03640955	0.02209473	0.03537256	0.01630736	0.02950931	0.03031892
Overhead	1.50542	1.43764	1.47914	1.38908	1.36926	1.4146	1.2705	1.15324	1.09446
95% CI	0.48809795	0.4061156	0.18518373	0.1327238	0.2746462	0.14178926	0.11419132	0.10765204	0.16597948
Latency	2576.98	2558.28	2722.58	2583.52	2632.06	2684.84	2679.02	2630.48	2620.62
95% CI	332.926249	224.982135	144.104795	256.100963	29.9733266	141.354393	142.220292	95.5210879	149.317918

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.07482	0.12634	0.19756	0.25362	0.31698	0.35606	0.35638	0.38826	0.40126
95% CI	0.01359141	0.0057828	0.01779952	0.03157788	0.01980684	0.04102084	0.01458802	0.02090169	0.01431087
Overhead	2.3835	2.54494	2.64286	2.4963	2.30624	2.21316	2.12096	2.08726	1.8962
95% CI	0.525234	0.43482879	0.22963393	0.21604837	0.10808959	0.16530469	0.26846484	0.12342172	0.15186171
Latency	2789.72	2694.14	2835.06	2748.6	2899.16	2708.42	2748.76	2724.1	2648.78
95% CI	337.883139	104.713099	161.505821	84.5909514	118.741584	95.9314909	107.645454	68.2858306	162.130234

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.06766	0.1325	0.21062	0.28008	0.33984	0.37876	0.3979	0.42382	0.41484
95% CI	0.00714258	0.01196206	0.00459476	0.01922702	0.00682107	0.00507086	0.01035848	0.01388698	0.01481521
Overhead	3.68042	3.44014	3.21322	3.56808	3.49326	3.29098	3.22706	3.05636	2.67036
95% CI	0.43515014	0.62025665	0.39817451	0.22899593	0.28489332	0.28156521	0.23584669	0.19243351	0.14484779
Latency	2739.72	2700.38	2797.18	2870.96	2916.28	2910.04	2766.66	2752.5	2665.3
95% CI	149.704075	210.621315	127.319803	88.130844	122.528229	74.0694611	146.718238	111.832727	147.596762

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.07392	0.13258	0.226	0.32282	0.38938	0.43982	0.46108	0.47406	0.47776
95% CI	0.00705606	0.0059824	0.0169036	0.01559589	0.02841489	0.013226	0.01811611	0.02267016	0.02102128
Overhead	5.20162	5.09898	5.53048	5.53586	5.69426	5.44456	5.07062	4.98602	4.79908
95% CI	0.75886356	0.38712162	0.28554307	0.34581315	0.20047828	0.18918151	0.12780846	0.19401495	0.20822752
Latency	2769.2	2866.48	2947.86	3005.06	3003.8	2936.12	2912.98	2872.08	2908.32
95% CI	455.411163	74.2703747	278.510893	173.270758	103.702124	106.863793	82.9268483	67.9940647	62.1839115

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.08174	0.14512	0.25736	0.35474	0.4307	0.47486	0.51522	0.5284	0.5345
95% CI	0.00787242	0.00750705	0.01577854	0.01366837	0.01156868	0.00735736	0.00681238	0.01617045	0.01508241
Overhead	6.52332	7.29124	7.95058	8.33006	8.44522	7.82612	7.51458	7.362	6.70562
95% CI	1.03024747	0.77585911	0.4608486	0.49128815	0.30690307	0.22868229	0.11555375	0.31063477	0.2625538
Latency	2783.38	2921.14	3041.9	3020.6	3116.96	2959.14	3005.12	2978.7	2910.1
95% CI	203.161038	151.19158	119.261371	137.064949	74.2580515	74.627977	82.3311959	97.7098212	140.885809

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.084	0.15728	0.28142	0.39466	0.46992	0.5358	0.5652	0.57302	0.57514
95% CI	0.00476644	0.00583048	0.0094201	0.01232181	0.01928387	0.02077479	0.0106465	0.0190362	0.01060925
Overhead	8.16534	9.16808	10.89334	11.12858	10.51216	10.4593	9.8445	9.80352	9.09548
95% CI	1.19066437	0.33541838	0.52512822	0.37854847	0.26708896	0.10169384	0.2453682	0.33649968	0.50760297
Latency	2756.24	3064	3211.68	3267.4	3108.7	3023.98	2983.8	2985.62	2941.54
95% CI	301.07205	197.823878	150.710379	64.404239	115.08871	114.417337	121.720924	109.603121	90.7107609

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E:
Random Mobility Simulations, 1000m Duration, Aggregate Data

This appendix includes the means and confidence intervals of the performance measurements for the 1000m duration random mobility simulations.

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.10492	0.12676	0.23814	0.32018	0.41516	0.51476	0.50704	0.51228	0.49024
	0.02909251	0.03011984	0.03892182	0.02547657	0.05510027	0.08993414	0.03420943	0.04833514	0.06972928
Overhead	3.42062	3.26844	2.72656	2.0968	1.59134	1.55414	1.39664	1.34214	1.53054
	0.58711812	0.29629753	0.23914923	0.45665087	0.2556554	0.16147888	0.17483781	0.24617478	0.22945344
Latency	2654.66	2616.42	2521.6	2554.9	2377	2220.66	2207.42	2177.94	2245.78
	236.267152	227.945879	212.780384	293.621797	126.997677	212.201278	77.2884924	72.5423847	165.397158

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.11704	0.2058	0.36888	0.47804	0.60078	0.6459	0.69124	0.71524	0.7229
	0.01312663	0.02508714	0.06013958	0.02434827	0.03299013	0.03433757	0.01439891	0.01063947	0.01585494
Overhead	9.26532	9.15108	8.67134	7.78478	7.0841	6.8993	6.00632	5.71758	5.54748
	0.56171776	0.89313455	0.18354481	0.58548289	0.32677835	0.40574008	0.35161721	0.46253397	0.58118277
Latency	3058.84	3013.7	2902.56	2674.56	2409.56	2235.9	2237.5	2176.48	2189.92
	277.697058	188.449869	107.853961	103.232792	66.6712059	128.482558	82.8616541	160.717623	217.639303

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.13612	0.27616	0.4562	0.57546	0.65818	0.71274	0.73378	0.75746	0.78424
	0.01561695	0.01803548	0.02752842	0.02419311	0.0118728	0.01402053	0.01427411	0.00801263	0.00917051
Overhead	14.3911	14.50998	14.98158	14.9481	14.9588	14.15366	13.86286	13.2741	12.9148
	1.41474443	0.63273554	0.74597613	0.87884027	0.97777712	0.47289791	0.66038939	0.55315728	0.6375871
Latency	3149.24	3304	3114.58	2695.5	2330.44	2182.84	2071.5	1940.9	1901.88
	150.403714	116.600282	197.267712	61.0005272	108.755471	46.4664944	62.7621542	86.5339595	79.7415819

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.1658	0.32172	0.51758	0.63032	0.6846	0.7151	0.74024	0.76962	0.78078
	0.02453756	0.01534307	0.02419866	0.02092976	0.01034463	0.01402064	0.00864389	0.01431537	0.01025986
Overhead	20.36648	21.65274	24.7307	25.58912	26.6432	26.82178	25.79146	23.80026	23.72766
	1.3965431	0.86958421	0.81503532	0.9736253	1.10437545	1.39395647	1.0463919	1.42246935	0.78942687
Latency	3462.32	3434.66	3001.88	2585.66	2246.06	1939.42	1880.92	1786.2	1741.56
	142.904344	146.298968	110.544168	90.3345162	136.772702	50.761238	103.617898	94.0757362	63.4400756

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.2075	0.4254	0.6137	0.66626	0.6886	0.6985	0.7206	0.74698	0.75996
	0.02489699	0.02081297	0.02104392	0.00954249	0.01167824	0.01280239	0.00625594	0.01212305	0.01029107
Overhead	31.73324	36.47512	46.29914	56.76622	60.77678	62.11636	60.94234	57.81596	56.27386
	1.82397775	1.40269936	0.69786721	3.07148423	1.86406621	1.41445227	1.89756583	1.45376561	2.62238258
Latency	3624.86	3577.68	2983.98	2320.5	1949.84	1707.58	1609.5	1540.66	1475.02
	174.936723	90.8776925	63.5680067	143.548898	65.0689158	49.6370151	89.1432171	63.3183874	38.9226907

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.27266	0.49628	0.65556	0.67752	0.67152	0.6822	0.69142	0.71036	0.721
	0.01860604	0.01839411	0.00916925	0.00930726	0.00649736	0.00768128	0.01054808	0.01237641	0.00857692
Overhead	45.40654	56.37804	75.83106	99.2795	111.7623	115.36274	114.46014	108.55106	106.49974
	1.75416146	1.01599956	1.5984623	1.60822432	1.82900381	2.89542718	3.13306087	2.64477316	1.41037463
Latency	3743.74	3468.9	2884.98	2182.56	1792.68	1560.9	1403.28	1372.56	1331.36
	98.7452216	47.4305083	69.6365643	67.4587377	62.4252919	44.1846618	56.007952	43.5154607	14.4074763

Epidemic, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.30036	0.54414	0.66076	0.66814	0.6445	0.65882	0.6739	0.68696	0.6858
	0.02238865	0.01015344	0.00588095	0.01769548	0.01236594	0.00687942	0.00988313	0.01102238	0.00667387
Overhead	58.56264	78.7409	117.91818	153.53834	183.80442	184.6216	181.9282	174.51156	172.31932
	1.36747398	3.66987327	4.03761859	4.97640341	4.06902614	3.74625493	2.95546065	4.72530726	3.53340821
Latency	3936.5	3500.4	2720.04	2107.18	1614.04	1416.36	1306.46	1251.6	1201.96
	86.1687977	84.2343886	116.090325	110.279701	40.5460431	54.3484147	31.6371747	34.2120682	34.2769662

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.10572	0.12268	0.23996	0.34012	0.44104	0.55182	0.54138	0.53508	0.51728
	0.03068862	0.02775048	0.03958889	0.02629711	0.06796393	0.1040469	0.04230654	0.0486769	0.07787097
Overhead	2.55462	2.60712	2.16754	1.59244	1.20428	1.09598	0.99628	0.98918	1.11338
	0.18858482	0.11348484	0.16390775	0.15622279	0.15219641	0.04743956	0.14992488	0.19150868	0.16951195
Latency	2735.04	2691.64	2679.88	2761.44	2606.34	2445.2	2465.76	2409.02	2478.8
	241.891342	335.103509	231.078397	215.581918	73.4719248	186.360592	95.5986778	124.174452	174.209486

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.11932	0.21658	0.40392	0.56004	0.73876	0.79788	0.83728	0.85034	0.84754
	0.0178466	0.028679	0.07258954	0.04184643	0.05060635	0.04283816	0.01929727	0.01737988	0.02967091
Overhead	7.1019	6.79964	6.33968	5.48134	4.53802	4.10032	3.6144	3.33132	3.08216
	0.41030888	0.20329727	0.28630052	0.10073418	0.15700807	0.16850397	0.11499669	0.28324653	0.13162747
Latency	3107.84	3107.28	3027.86	2917.72	2647.86	2423.16	2358.8	2259.24	2270.7
	278.870804	185.268177	53.3858197	60.3218338	30.590151	178.215134	92.3762133	186.158514	192.235634

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.14044	0.29242	0.52894	0.72886	0.8835	0.93056	0.95308	0.95854	0.96672
	0.01910501	0.01802625	0.0437787	0.04433328	0.01560846	0.01480827	0.0090864	0.00672485	0.01133368
Overhead	11.62036	11.34206	10.82478	9.73932	8.52276	7.61578	7.10492	6.51248	6.0313
	0.69039508	0.3378448	0.31586936	0.24055074	0.15840386	0.16291411	0.29002716	0.1608516	0.04426833
Latency	3201.58	3379.8	3266.2	2916.16	2436.7	2109.2	1953.3	1757.26	1710.88
	194.023526	67.3469098	125.910025	79.2029619	101.586442	81.9217405	100.75914	108.320244	92.6491203

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.16978	0.355	0.65382	0.85486	0.95696	0.97148	0.97382	0.9779	0.97944
	0.02628025	0.02242037	0.04172004	0.03689963	0.01116759	0.008121	0.0058841	0.0038018	0.00320508
Overhead	16.47846	16.1992	16.31822	15.36592	13.45074	11.74824	11.06346	10.28734	9.58
	0.61144416	0.55633785	0.38136637	0.43623742	0.49792665	0.31536816	0.08825933	0.28504559	0.09190499
Latency	3470	3498.04	3173.4	2651.02	2099.42	1663.8	1556.42	1430.26	1368.06
	126.229971	121.87002	118.231941	87.1359949	153.37801	38.1577913	69.6448108	43.9567119	64.6819115

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.21786	0.48524	0.81314	0.95752	0.98162	0.98272	0.98382	0.98592	0.98468
	0.02988268	0.02968961	0.01940368	0.00609517	0.00243379	0.00409128	0.00158136	0.00219392	0.00302798
Overhead	26.08668	27.34644	28.67726	26.48996	21.43746	19.93262	19.54166	18.85658	17.8185
	1.67200772	0.94297774	0.81274333	0.94531841	0.73093838	0.21054255	0.31482194	0.15881221	0.20251884
Latency	3664.6	3588.72	2928.1	2044.86	1513.46	1230.24	1111.9	1047.16	970.74
	149.501399	153.256913	102.655392	103.420199	81.6579646	30.2755521	33.7251805	44.6443912	23.4361572

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.29058	0.59726	0.9014	0.97476	0.98142	0.98708	0.98828	0.98944	0.98884
	0.02434824	0.01992466	0.00750308	0.00329287	0.00255885	0.00193237	0.00197186	0.0007272	0.00092752
Overhead	36.80082	40.22256	44.28296	35.50696	29.24966	28.61072	28.19482	27.59794	26.54962
	1.92954348	0.64229035	0.83724387	1.0049982	0.36039505	0.24555684	0.08370065	0.5179168	0.14435017
Latency	3760.88	3468.48	2652.7	1682.32	1242.52	1014.82	883.62	825.04	788.94
	75.4466658	87.8988669	34.455337	35.397051	22.9559127	26.4660033	8.66695702	3.06994989	15.4177974

GAPR, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.32226	0.67518	0.9465	0.97956	0.9864	0.98748	0.99028	0.99102	0.9914
	0.03022765	0.02564621	0.00393335	0.00213732	0.00188717	0.00228345	0.00107675	0.00184252	0.00136017
Overhead	48.07434	56.39084	60.06802	41.4428	38.13438	37.62378	37.15622	36.31838	35.52448
	1.48087258	1.41515722	0.77548871	1.08240304	0.40017447	0.35713264	0.38832097	0.37772324	0.31047248
Latency	3945.18	3380.38	2282.62	1433.1	1041.52	865.68	765.78	694.46	661.66
	99.3074451	59.1573348	85.0959092	53.8857141	9.01314651	17.6297487	6.84797676	12.1919139	15.1840328

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.10234	0.12246	0.2286	0.31126	0.40556	0.50352	0.5035	0.50108	0.48092
	0.02687143	0.03000534	0.03916028	0.02835084	0.05994683	0.08707586	0.04129219	0.04400989	0.06879773
Overhead	0.61818	0.55246	0.55644	0.29584	0.27442	0.25238	0.2404	0.19614	0.24048
	0.27695255	0.23262401	0.2173173	0.13508638	0.06860749	0.10553565	0.07440329	0.06247055	0.06931252
Latency	2722.6	2630.06	2591.94	2661.88	2512.68	2339	2352.86	2317.08	2368.72
	224.056144	288.63273	229.695668	226.440255	111.586479	175.169374	111.774887	175.710579	191.717775

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09696	0.1617	0.27248	0.38522	0.53558	0.58688	0.61478	0.64544	0.66038
	0.01165995	0.0245942	0.04191526	0.02759672	0.04819859	0.03726403	0.01697028	0.01753905	0.02863717
Overhead	1.0834	1.1619	1.18852	1.11412	1.07578	1.07694	0.88302	0.85632	0.74404
	0.32457514	0.35711408	0.25881433	0.15884223	0.08439742	0.10143713	0.05800959	0.11756324	0.02676954
Latency	2884.88	2858.96	2771.62	2698.86	2586.52	2456.96	2425.04	2346.7	2360.76
	349.691992	276.200483	120.058809	75.4988686	37.5094959	134.238023	80.0099801	218.430473	170.192634

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09714	0.17264	0.29756	0.43778	0.56864	0.63588	0.67202	0.71336	0.74846
	0.01130821	0.01116621	0.0229541	0.02615338	0.02929296	0.01300829	0.01769609	0.01272345	0.02110365
Overhead	2.02614	1.87476	1.8974	2.1182	1.81718	1.74054	1.65392	1.5139	1.48796
	0.30372328	0.34191777	0.27222906	0.14098954	0.09254868	0.17147192	0.0651039	0.07050356	0.09123018
Latency	2747.82	2836.98	2919.42	2851.36	2740.52	2542.44	2414.72	2270.38	2222.76
	110.791272	156.1603	155.70069	90.1953566	150.848134	46.8675612	108.567361	134.013153	58.4969113

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.0982	0.1768	0.32906	0.47908	0.61916	0.69946	0.72766	0.76852	0.7861
	0.01824253	0.01675211	0.02643237	0.03054006	0.02252082	0.01605943	0.01745843	0.01016171	0.00925061
Overhead	2.73772	2.66398	2.74666	2.79082	2.6318	2.64976	2.53194	2.39154	2.1503
	0.35396318	0.30677048	0.21163258	0.25776024	0.24025622	0.15006324	0.11112489	0.11687523	0.11922512
Latency	2916.16	2975.38	2953.92	2876.7	2741.56	2471.8	2385.38	2259.48	2226.56
	174.221933	129.758979	102.829849	65.2451013	151.64293	22.140126	64.0260763	62.5156183	102.111355

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.10236	0.1989	0.36848	0.55252	0.6848	0.77856	0.81784	0.84022	0.86342
	0.01420677	0.01236719	0.03381961	0.0230673	0.02204347	0.01847382	0.01333954	0.01501346	0.0091207
Overhead	4.11686	4.2554	4.41296	4.61636	4.46554	4.49472	4.33272	4.01448	3.73894
	0.30557526	0.41290406	0.32605185	0.18472655	0.20669265	0.08235824	0.2419673	0.09556064	0.09461547
Latency	2950.2	3069.84	3015.78	2903.62	2648.56	2413.06	2298.18	2167.26	2033.02
	231.495191	115.749647	91.1908337	54.6698468	113.949332	70.6269244	71.2419475	84.3486399	83.2920297

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.108	0.21588	0.41836	0.61562	0.7514	0.82878	0.87634	0.89302	0.89902
	0.0103744	0.01633163	0.01647314	0.01728956	0.0131335	0.01671282	0.01336581	0.00585718	0.01142378
Overhead	5.84742	6.31764	6.95884	6.52674	6.65402	6.45376	6.17596	6.12596	5.43902
	0.5753362	0.97088302	0.20422368	0.22490989	0.39191184	0.31790673	0.30423578	0.16377865	0.23367665
Latency	3005.36	3054.82	3157.14	2915.12	2565.16	2361.7	2135.24	1995.38	1944.74
	152.049313	102.883285	41.9366018	65.8331025	88.6533574	61.7585874	72.0352736	74.5542153	55.469496

GAPR2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.10722	0.23794	0.46258	0.66634	0.8146	0.87814	0.9091	0.926	0.9314
	0.00840598	0.01434125	0.01673402	0.01382508	0.01344356	0.01715106	0.01103894	0.01091959	0.00986361
Overhead	7.71394	7.90262	8.5333	8.74088	9.07472	8.95608	8.63056	8.01904	7.26924
	0.83588877	0.63335593	0.22554552	0.44695404	0.2946748	0.43030297	0.36375539	0.4086869	0.36883049
Latency	3089.06	3334.16	3186.9	2903.96	2471.92	2208.6	1998.22	1883.86	1810.8
	386.674081	118.286878	122.048339	57.0394683	25.6681411	38.1906054	44.043504	26.7981757	32.7796944

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.1055	0.12166	0.2376	0.34112	0.44042	0.55174	0.5434	0.53618	0.51922
	0.03072924	0.03178885	0.04375382	0.0227637	0.06747175	0.10755067	0.04458542	0.04686105	0.07802895
Overhead	2.53244	2.5949	2.17232	1.57432	1.20964	1.09496	0.9906	0.98582	1.1087
	0.23772614	0.26133345	0.17653393	0.16325442	0.14866106	0.05779653	0.14474908	0.18144124	0.1610595
Latency	2727.9	2658.38	2653.3	2798.82	2613.2	2442.96	2462.6	2407.08	2503.06
	222.777637	285.567784	181.675238	190.324399	72.0214356	183.182867	69.5441585	152.120396	171.130076

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.1186	0.21674	0.40008	0.55894	0.73668	0.80062	0.83698	0.84854	0.84738
	0.01519657	0.02595889	0.0724057	0.04138697	0.04915824	0.04396643	0.01929108	0.02038856	0.02875135
Overhead	7.07368	6.66706	6.29398	5.42702	4.54	4.0842	3.59798	3.34096	3.08164
	0.36233306	0.29896588	0.30088882	0.09541582	0.14108743	0.15744894	0.12119593	0.29546008	0.12277887
Latency	3111.26	3104.76	3040.58	2915.82	2642.18	2427.74	2359.28	2255.16	2283.5
	321.629273	223.971296	68.826467	84.1241975	47.5006997	171.940621	99.1874435	176.388884	195.792421

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.13894	0.29012	0.52266	0.72784	0.88498	0.92956	0.95296	0.95802	0.966
	0.01802907	0.01670313	0.03442832	0.04358812	0.01189712	0.01300331	0.00913303	0.00660969	0.01103091
Overhead	11.69392	11.21036	10.6526	9.67824	8.50592	7.63832	7.09122	6.49286	6.03636
	0.4645765	0.32236335	0.23933374	0.31953386	0.19679654	0.17747613	0.28237534	0.16046819	0.02465955
Latency	3196.68	3399.28	3283.02	2913.36	2454.3	2104.4	1957.28	1753.68	1712.04
	186.600833	33.7855154	150.720248	93.0978353	115.545792	81.0597877	112.72666	109.96687	90.1784327

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.17016	0.35	0.64898	0.86008	0.9575	0.97092	0.97422	0.97788	0.97944
	0.02614825	0.0235047	0.03688258	0.03580574	0.01164751	0.00696406	0.0057516	0.00396595	0.00312717
Overhead	16.33946	16.1131	16.0605	15.2208	13.433	11.72756	11.02598	10.27898	9.58708
	0.61641669	0.63328204	0.32465293	0.4478656	0.52268494	0.3005618	0.08904127	0.27551932	0.10006508
Latency	3475.82	3520.38	3178.34	2668.24	2097.4	1664.32	1557.06	1432.5	1370.16
	122.994243	126.06441	119.636585	77.3942527	162.42453	35.0562231	67.815807	40.835269	60.3992389

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.21924	0.48274	0.81572	0.95886	0.98178	0.98272	0.98382	0.98584	0.98468
	0.02910273	0.03416377	0.01953094	0.0050996	0.0026404	0.00409128	0.00158136	0.00212103	0.0027561
Overhead	25.70042	26.91054	28.27608	26.19876	21.33968	19.9342	19.54256	18.8596	17.80942
	1.90616365	0.82179521	0.60347255	0.66635398	0.72071096	0.21905165	0.32701076	0.185008	0.19411141
Latency	3690.1	3614.64	2938.9	2040.18	1513.28	1230.1	1112.12	1045.66	969.3
	155.659359	90.1799286	66.962883	97.4963	86.7679919	30.9181815	32.7963164	44.6540596	21.9567486

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.28902	0.60256	0.9076	0.97484	0.9815	0.98708	0.98828	0.98944	0.98884
	0.02197349	0.01576904	0.00901554	0.00314438	0.00238677	0.00193237	0.00197186	0.0007272	0.00092752
Overhead	36.60712	39.28236	44.28554	35.23344	29.23286	28.6219	28.21584	27.59152	26.5543
	2.06145992	0.37771444	0.77796921	0.74476225	0.35864074	0.24198599	0.09454758	0.50259445	0.12893492
Latency	3770.74	3486.98	2677.54	1682.56	1243.44	1014.84	884.26	825.08	788.84
	91.1843329	67.4848895	38.0429315	33.5706085	23.6442626	24.238	10.6893497	3.51349228	17.8416791

MaxProp, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.32398	0.68432	0.95134	0.97964	0.9864	0.98748	0.99028	0.99102	0.99116
	0.03042285	0.02284432	0.00580775	0.00201096	0.00188717	0.00228345	0.00107675	0.00184252	0.00119613
Overhead	47.38642	55.33284	59.8352	41.17618	38.13802	37.62624	37.17452	36.31402	35.55798
	1.32231976	1.51977022	0.66823969	0.98528252	0.39037389	0.35872313	0.38807417	0.39044937	0.28769827
Latency	3960.86	3448.98	2267.72	1434.62	1040.86	865.9	765.6	694.34	661.62
	114.387874	62.7011674	84.3616072	52.8165977	8.64210631	17.985922	6.34525535	12.2159167	15.0928332

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.10298	0.12338	0.22806	0.32706	0.40612	0.50612	0.5046	0.50154	0.48344
	0.02794839	0.0277295	0.04003501	0.01928353	0.06030285	0.08696673	0.03723833	0.04372452	0.07151864
Overhead	1.71002	1.50146	1.34022	1.08826	0.9968	0.89242	0.80708	0.76276	0.8274
	0.32433387	0.1891442	0.13830054	0.11132619	0.11020976	0.10598846	0.1217788	0.11188535	0.14574428
Latency	2651.82	2612.06	2505.2	2607.14	2418.9	2230.2	2225.52	2161.76	2276.54
	210.684466	237.770658	218.774907	270.880727	80.9833402	183.40157	77.6610657	129.239556	125.959992

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.10964	0.18636	0.33698	0.46208	0.60744	0.67078	0.71104	0.7243	0.73888
	0.0144147	0.02477946	0.05227349	0.02559657	0.03517829	0.03763854	0.013542	0.02957338	0.02576855
Overhead	3.9364	3.64172	3.64126	3.57148	3.5055	3.33698	3.03536	2.82824	2.59088
	0.45164926	0.14028884	0.18505665	0.0344773	0.11200343	0.21987102	0.123156	0.21906105	0.23562279
Latency	3027.16	2954.08	2905.12	2798.02	2538.88	2390.58	2354.42	2295.76	2342.48
	266.932544	274.974663	103.142727	71.171797	102.523498	185.017798	98.2512253	163.519856	177.353793

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.11706	0.22728	0.40054	0.57818	0.70124	0.76686	0.79844	0.8182	0.84638
	0.01430088	0.02166595	0.0299682	0.0299692	0.02380785	0.02364622	0.02294201	0.00715601	0.01806704
Overhead	6.27714	5.70384	5.60046	5.6217	6.13918	6.13756	5.9998	5.86486	5.47854
	0.56310053	0.35703398	0.23942747	0.31763635	0.14808186	0.23642317	0.15369922	0.09333175	0.20022256
Latency	2958.54	3157.72	3121.46	2918.64	2565.72	2343.18	2184.54	2048.8	2056.24
	180.326927	140.931938	85.2556557	95.2751853	117.197014	57.4300405	52.4311351	120.666617	47.0087988

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.12954	0.25188	0.48086	0.65194	0.76428	0.81998	0.83868	0.85538	0.87946
	0.01872868	0.01701882	0.02636857	0.041683	0.02018351	0.01778755	0.01212433	0.01541675	0.00360681
Overhead	8.23676	7.76702	7.61998	8.30134	9.22372	9.78198	9.8618	9.50212	8.9968
	0.27357778	0.29567995	0.13806004	0.29517088	0.26920274	0.56077509	0.27525415	0.39837114	0.23624435
Latency	3231.78	3335.72	3139.82	2768.24	2439.74	2038.5	1975.4	1886.74	1868.26
	201.481217	48.6674794	150.707793	89.6405132	117.031644	30.4734297	61.1778827	93.3103514	75.5011659

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.14414	0.31294	0.58858	0.75708	0.8169	0.84976	0.86398	0.87964	0.8964
	0.01435817	0.01641642	0.02448022	0.00595895	0.00791457	0.00638317	0.01081439	0.00983528	0.01014486
Overhead	12.09378	11.56364	11.98318	14.00832	16.86552	18.9183	19.7203	19.71128	18.81728
	0.66034339	0.44406515	0.19565296	0.43791787	0.5615914	0.604319	0.51841798	0.55380156	0.66855881
Latency	3351.14	3420.92	3082.58	2524.12	2084.16	1795.78	1648.9	1540.26	1494.64
	166.030085	67.9411151	108.007833	89.232475	109.163968	52.2344742	53.7265444	45.5246077	52.5276412

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.17428	0.38052	0.6644	0.80398	0.84392	0.86972	0.87536	0.88714	0.9005
	0.01857074	0.02314037	0.0162005	0.00625187	0.00540545	0.00992997	0.00474	0.00934413	0.0115174
Overhead	16.14368	15.14962	16.89952	21.4468	26.27224	30.14026	32.58684	33.05288	31.53556
	1.37459072	0.53743554	0.31670832	0.77260875	0.60525854	1.05103013	0.59627177	0.77111489	0.94727236
Latency	3440.42	3460.16	3028.04	2312.18	1873.24	1598.96	1387.44	1313.78	1275.3
	141.234747	65.2907263	39.8691717	47.6004003	38.2433116	27.8558677	42.2073172	37.9833927	21.3468729

PRoPHETv2, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.1749	0.43904	0.71772	0.82628	0.85288	0.86452	0.87642	0.8908	0.9023
	0.01197095	0.01750737	0.01361885	0.01074646	0.0058841	0.00463375	0.00684178	0.00746704	0.00892574
Overhead	20.32436	19.1133	22.43584	29.31434	38.68774	43.72116	48.11312	48.53754	47.09792
	0.70620654	0.92339437	0.40721252	0.54552018	0.60542134	1.40814606	0.70725035	1.05801315	1.22838535
Latency	3649.68	3588.56	2854.42	2160.2	1659.3	1394.98	1251.26	1168.9	1141.84
	119.233574	98.9125315	109.670334	84.6740653	27.165639	40.9018897	5.29610088	7.98342639	25.4840155

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	5	5	5	5	5	5	5	5	5
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.10156	0.12128	0.22974	0.3104	0.40494	0.50382	0.50212	0.4996	0.47912
	0.02778244	0.02843253	0.03681032	0.02870353	0.06119134	0.0857813	0.04111477	0.04396569	0.06801907
Overhead	0.59022	0.52818	0.53856	0.29864	0.27572	0.24818	0.24272	0.1917	0.22826
	0.27202554	0.23718396	0.1974648	0.13657064	0.07479572	0.09757581	0.06103757	0.06900821	0.06888587
Latency	2702.94	2639.5	2584.48	2662.28	2522.04	2341.9	2354.62	2307.78	2365
	229.25063	283.83058	214.084666	223.608094	114.283371	191.2825	123.926944	181.609424	195.654513

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	10	10	10	10	10	10	10	10	10
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09696	0.16162	0.27224	0.38562	0.53386	0.5843	0.61236	0.643	0.65964
	0.01165995	0.02454751	0.04181096	0.02857627	0.047089	0.03719473	0.01757681	0.01543913	0.02857293
Overhead	1.0834	1.16026	1.19162	1.12166	1.07164	1.07086	0.87124	0.85608	0.74958
	0.32457514	0.35555843	0.26422416	0.16894306	0.08826007	0.10385904	0.05793361	0.13670448	0.04440077
Latency	2884.88	2858.96	2772.76	2706.26	2579.1	2466.5	2413.82	2341.76	2359.46
	349.691992	276.200483	122.714998	79.9971169	17.2023126	130.910756	79.9113476	227.619001	160.134696

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	15	15	15	15	15	15	15	15	15
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09714	0.1725	0.29772	0.4362	0.56724	0.63362	0.66464	0.70914	0.73768
	0.01130821	0.01096362	0.02324987	0.02781535	0.02909691	0.01174124	0.0135988	0.01482258	0.01976341
Overhead	2.02844	1.87484	1.90494	2.11612	1.8064	1.72428	1.63886	1.50142	1.43566
	0.3065476	0.33867748	0.26320888	0.14015214	0.11277802	0.17429209	0.06428107	0.08799296	0.08975748
Latency	2747.82	2835.24	2916.48	2851	2747.12	2545.88	2402.56	2269.62	2210.72
	110.791272	152.889169	155.510086	94.1324222	142.545661	48.7171106	105.921712	146.566178	75.0126618

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	20	20	20	20	20	20	20	20	20
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.09812	0.17664	0.32992	0.47804	0.6164	0.6922	0.72124	0.76076	0.77314
	0.01825487	0.01656368	0.02796411	0.03154811	0.02490194	0.01928161	0.01606254	0.00931231	0.0045871
Overhead	2.7395	2.6665	2.75014	2.78912	2.6089	2.63928	2.5104	2.36706	2.1337
	0.35834882	0.30930884	0.21140004	0.26072471	0.24392209	0.13989728	0.12969824	0.1726972	0.13846717
Latency	2913.66	2972.12	2957.82	2881.98	2750.06	2468.96	2387.52	2258.96	2223.24
	172.002681	130.693391	100.752576	55.6891079	146.361235	33.915456	78.4725458	77.7916418	119.804478

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	30	30	30	30	30	30	30	30	30
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.10236	0.19828	0.36778	0.55018	0.6809	0.7714	0.80938	0.82448	0.84856
	0.01420677	0.01150615	0.0320785	0.01916982	0.0209013	0.01817183	0.01179038	0.01920919	0.01192275
Overhead	4.1178	4.27398	4.40532	4.58784	4.43902	4.45892	4.28594	3.9526	3.6702
	0.30172337	0.41445166	0.34167026	0.17137468	0.20417818	0.08836712	0.22722598	0.07801015	0.11239683
Latency	2950.2	3063.94	3026.2	2900.6	2643.64	2415.06	2300.08	2181.92	2026.74
	231.495191	113.570303	85.7556658	73.1640436	105.48433	77.7038945	62.8257689	82.3188302	72.1120139

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	40	40	40	40	40	40	40	40	40
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.10816	0.2165	0.41804	0.61392	0.74232	0.82124	0.86354	0.88012	0.88252
	0.01032733	0.01566491	0.01312251	0.01929228	0.01469639	0.01621991	0.0168351	0.00726633	0.01107568
Overhead	5.85592	6.29356	6.98928	6.5245	6.64194	6.44812	6.05218	5.93618	5.39334
	0.57813487	0.96362291	0.29654393	0.17839575	0.31727338	0.3495857	0.28062599	0.18900684	0.21108945
Latency	3002.2	3062.44	3155.8	2924.96	2559.88	2366.26	2136.12	2008.02	1979.14
	155.490073	82.0019948	36.6602215	64.6014761	68.4042715	51.8298128	76.6437357	91.7933818	52.6758738

Vector, 35M Buffer, 2MBps Transmit Speed									
# Nodes	50	50	50	50	50	50	50	50	50
Speed	1	2	4	7	12	18	25	35	50
D-Ratio	0.10698	0.23762	0.4628	0.66062	0.81032	0.87196	0.90138	0.91772	0.91954
	0.00871043	0.01375767	0.01784328	0.01503321	0.01176027	0.01535593	0.0115436	0.00944375	0.01343197
Overhead	7.73374	7.90704	8.54056	8.64672	8.93286	8.87128	8.54938	7.8724	7.1336
	0.84749313	0.67807884	0.19509931	0.40582707	0.28114712	0.39716044	0.36392856	0.40389448	0.30716741
Latency	3087.36	3324.1	3212.28	2895.88	2479.08	2227.82	2020.52	1922.36	1835.66
	384.785649	104.781545	139.123711	51.5144232	43.1613908	33.7207688	66.4609514	34.685301	38.5561764

THIS PAGE INTENTIONALLY LEFT BLANK

REFERENCES

[1] H. Kang and D. Kim. Vector routing for delay tolerant networks. In *IEEE 68th Vehicular Technology Conference (VTC)*, Fall 2008.

[2] J. P. Rohrer et al. Geolocation assisted predictive routing (GAPR) protocol for heterogeneous DTN mobility patterns. Naval Postgraduate School (internal), 2012.

[3] A. Keränen et al. The ONE simulator for DTN protocol evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST, New York, NY, USA, 2009. ISBN 978-963-9799-45-5.

[4] The network simulator: ns-2. `http://www.isi.edu/nsnam/ns/`, July 2015.

[5] Bold alligator. `http://www.public.navy.mil/usff/ba/Pages/about.aspx`.

[6] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*.

[7] K. Fall and S. Farrell. DTN: an architectural retrospective. *Selected Areas in Communications, IEEE Journal on*, 26(5):828–836, June 2008. ISSN 0733-8716.

[8] S. Grasic and A Lindgren. An analysis of evaluation practices for DTN routing protocols. In *Proceedings of the seventh ACM international workshop on Challenged networks*, pp. 57–64. ACM, 2012.

[9] A. Balasubramanian et al. DTN routing as a resource allocation problem. *ACM SIGCOMM Computer Communication Review*, 37(4):373–384, 2007.

[10] P. Juang et al. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. *ACM Sigplan Notices*, 37(10):96–107, 2002.

[11] M. Ramadas. RFC 2488: Enhancing TCP over satellite channels using standard mechanisms, 1999. `https://tools.ietf.org/html/rfc2488`.

[12] M. Ramadas et al. RFC 2760: Ongoing TCP research related to satellites, 1999. `https://tools.ietf.org/html/rfc2760`.

[13] V. G. Cerf. TEDx mid-atlantic 2011, interplanetary internet, 2011. `https://www.youtube.com/watch?v=XTmYm3gMYOQ&feature=youtu.be`.

[14] M. Ramadaser et al. RFC 5326: Licklider transmission protocol - specification, 2008. `https://tools.ietf.org/html/rfc5326`.

[15] J. Jackson. The interplanetary internet, Aug 2005. `http://spectrum.ieee.org/telecom/internet/the-interplanetary-internet`.

[16] Interplanetary networking special interest group (ipnsig). `http://ipnsig.org/`.

[17] MANET routing, class notes for CS4554: Network modeling and analysis.

[18] S. Basagni et al. *Mobile ad hoc networking*. John Wiley & Sons, 2004.

[19] E. Royer et al. A review of current routing protocols for ad hoc mobile wireless networks. *Personal Communications, IEEE*, 6(2):46–55, 1999.

[20] M. Khabbaz et al. Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challenges. *Communications Surveys & Tutorials, IEEE*, 14(2):607–640, 2012.

[21] Y. Cao and Z. Sun. Routing in delay/disruption tolerant networks: A taxonomy, survey and challenges. *Communications Surveys & Tutorials, IEEE*, 15(2):654–677, 2013.

[22] J. Shen et al. Routing protocols in delay tolerant networks. *ITC-CSCC: 2008*, pp. 1577–1580, 2008.

[23] S. Ali et al. Routing protocols in delay tolerant networks-a survey. In *Emerging Technologies (ICET), 2010 6th International Conference on*, pp. 70–75. IEEE, 2010.

[24] A. Tovar et al. A DTN wireless sensor network for wildlife habitat monitoring. In *Electrical and Computer Engineering (CCECE), 2010 23rd Canadian Conference on*, pp. 1–5. IEEE, 2010.

[25] e learning - DTN, case b: Environmental deployment. `http://www.elearning-dtn.eu/Environment-deployment.htm`, 2010.

[26] H. Soroush et al. A retrospective look at the UMass DOME mobile testbed. *ACM SIGMOBILE Mobile Computing and Communications Review*, 15(4):2–15, 2012.

[27] A. Balasubramanian et al. Web search from a bus. In *Proceedings of the second ACM workshop on Challenged networks*, pp. 59–66. ACM, 2007.

[28] ExtremeCom. `http://extremecom.org/`, 2015.

[29] S. Saha et al. Post disaster management using delay tolerant network. In *Recent Trends in Wireless and Mobile Networks*.

[30] Y. Sasaki and Y. Shibata. Distributed disaster information system in DTN based mobile communication environment. In *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on*, pp. 274–277, Nov 2010.

[31] About multipeer connectivity. [online]. `https://developer.apple.com/library/\\prerelease/ios/documentation/MultipeerConnectivity/Reference/\\MultipeerConnectivityFramework/index.html`.

[32] J. Redi and R. Ramanathan. The DARPA WNaN network architecture. In *Military Communications Conference, 2011 - MILCOM 2011*, pp. 2258–2263, Nov 2011. ISSN 2155-7578.

[33] R. S. Mangrulkar and M. Atique. Routing protocol for delay tolerant network: A survey and comparison. In *Communication Control and Computing Technologies (ICCCCT), 2010 IEEE International Conference on*, pp. 210–215. IEEE, 2010.

[34] T. Spyropoulos et al. Efficient routing in intermittently connected mobile networks: The single-copy case. *IEEE/ACM Trans. Netw.*, 16(1):63–76, February 2008. ISSN 1063-6692. http://dx.doi.org/10.1109/TNET.2007.897962.

[35] E. Jones et al. Routing strategies for delay-tolerant networks. *Submitted to ACM Computer Communication Review (CCR)*, 2006.

[36] A. Lindgren et al. Probabilistic routing in intermittently connected networks. In *Service Assurance with Partial and Intermittent Resources*, pp. 239–254. Springer, 2004.

[37] S. Grasic et al. The evolution of a DTN routing protocol - PRoPHETv2. In *Proceedings of the 6th ACM Workshop on Challenged Networks*, CHANTS '11, pp. 27–30. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0870-0. http://doi.acm.org/10.1145/2030652.2030661.

[38] J. Burgess et al. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *INFOCOM*, volume 6, pp. 1–11, 2006.

[39] S. Lo et al. Routing and buffering strategies in delay-tolerant networks: Survey and evaluation. In *Parallel Processing Workshops (ICPPW), 2011 40th International Conference on*,
pp. 91–100. IEEE, 2011.

[40] P. Puri et al. A survey paper on routing in delay-tolerant networks. In *Information Systems and Computer Networks (ISCON), 2013 International Conference on*,
pp. 215–220. IEEE, 2013.

[41] L. Song and D. F. Kotz. Evaluating opportunistic routing protocols with large realistic contact traces. In *Proceedings of the second ACM workshop on Challenged networks*, pp. 35–42. ACM, 2007.

[42] K. Fall and K. Varadhan. The network simulator - ns-2, 2011. http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf.

[43] The ONE, the opportunistic network environment simulator. http://www.netlab.tkk.fi/tutkimus/dtn/theone/.

[44] A. Balasubramanian et al. DTN routing as a resource allocation problem. *ACM SIGCOMM Computer Communication Review*, 37(4):373–384, 2007.

[45] F. Ekman et al. Working day movement model. In *MobilityModels '08: Proceeding of the 1st ACM SIGMOBILE workshop on Mobility models*, pp. 33–40. ACM, New York, NY, USA, 2008. ISBN 978-1-60558-111-8.

[46] D. Broyles et al. Design and analysis of a 3–D gauss-markov mobility model for highly-dynamic airborne networks. In *Proceedings of the International Telemetering Conference (ITC)*. San Diego, CA, October 2010.

[47] QGIS Development Team. *QGIS Geographic Information System*. Open Source Geospatial Foundation, 2009. `http://qgis.osgeo.org`.

[48] Openstreetmap. [online]. `https://www.openstreetmap.org/`, 2015.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia
2. Dudly Knox Library
   Naval Postgraduate School
   Monterey, California